Flightplan: Dataplane Disaggregation and Placement for P4 programs

Nik Sultana  John Sonchack  Hans Giesen  Isaac Pedisich  Zhaoyang Han
Nishanth Shyamkumar  Shivani Burad  André DeHon  Boon Thau Loo

University of Pennsylvania

https://flightplan.cis.upenn.edu/

NSDI’21
Dataplane Programmability
Dataplane Programmability
Dataplane Programmability
Dataplane Programmability

Host (CPU) — NPU — FPGA

DPDK (Linux Foundation)

Netronome

Netcope
Dataplane Programmability

Host (CPU) — NPU — Switch — FPGA

DPDK
(Netronome)

(P4 Language Consortium)

FPGA
(Netcope)
Dataplane Programmability

Diagram showing connections between Host (CPU), NPU, Switch, and FPGA.
Dataplane Programmability

Research

Products
Dataplane Programmability

**Current paradigm:** dataplane program → one dataplane (device)
Dataplane Programmability

Current paradigm: dataplane program \(\rightarrow\) one dataplane (device)

- Conceptually simple.
- Familiar from programming other devices.
- Mismatch with Software-Defined Networking (the overarching vision).
Current paradigm: dataplane program mapped to one dataplane

Mismatch with Software-Defined Networking

- Target’s resources are dedicated to the program. Inefficient use of individual target resources.
- Program must entirely execute on a single target. Unnecessary constraints of program functionality.
- We’re meant to be “programming the network”. Programmability is scoped too conservatively.
Current paradigm: dataplane program → one dataplane

New paradigm: dataplane program → suitable mix of dataplanes
Current paradigm: dataplane program → one dataplane

New paradigm: dataplane program → suitable mix of dataplanes

- Resource-based program decomposition.
- Split into set of 1-1 dataplane programs.
**Current paradigm:** dataplane program ➝ one dataplane

**New paradigm:** dataplane program ➝ suitable mix of dataplanes

- Resource-based program decomposition.
- Split into set of 1-1 dataplane programs.

*Uses existing vendor toolchains, language & hardware.*
Example: “Crosspod” Program

Diagram showing decision-making process with paths for FEC-control, FEC-encoded, and compressed header conditions.
Example: “Crosspod” Program
Current paradigm: dataplane program → one dataplane

New paradigm: **Dataplane Disaggregation**

- Resource-based program decomposition.
- Split into set of 1-1 dataplane programs.
Dataplane Disaggregation
(≠ Server Disaggregation)

Virtual Dataplane → Set of Physical Dataplanes

“One big switch” → “One big programmable switch”
Current paradigm: dataplane program → one dataplane

How to implement?

Automated support needed

Dataplane Disaggregation:
 dataplane program → suitable mix of dataplanes

- Resource-based program decomposition.
- Split into set of 1-1 dataplane programs.

Uses existing vendor toolchains, language & hardware.

If manual: laborious, error-prone, hard to change.
Current paradigm: dataplane program → one dataplane

Dataplane Disaggregation:
 dataplane program → suitable mix of dataplanes

- Analyze program’s use of resources.
- Split into set of 1-1 dataplane programs.
Current paradigm: dataplane program → one dataplane

Dataplane Disaggregation:
 dataplane program → suitable mix of dataplanes

- Exploit heterogeneous resources.
- Meet resource & performance objectives.

- Analyze program’s use of resources.
- Split into set of 1-1 dataplane programs.
Heterogeneity
Heterogeneity

Programmable Switch

FPGA Boards

x86 Servers

DSL Testbed Deployment

Throughput

Power

Cost

~110W

~$5K

~90Gbps

26
Heterogeneity

- FPGA Boards: 
  - ~30W
  - ~$2K
  - ~200Gbps

- Programmable Switch:
  - ~110W
  - ~$10K
  - ~6Tbps

- x86 Servers:
  - ~110W
  - ~$5K
  - ~90Gbps

- DSL Testbed Deployment:
  - ~30W
  - ~$2K
  - ~200Gbps

Throughput

Power

Cost
Current paradigm: dataplane program $\rightarrow$ one dataplane

Dataplane Disaggregation:

- dataplane program $\rightarrow$ suitable mix of dataplanes

- Analyze program’s use of resources.
- Split into set of 1-1 dataplane programs.

- Exploit heterogeneous resources.
- Meet resource & performance objectives.
Current paradigm: dataplane program → one dataplane

Dataplane Disaggregation:

dataplane program → suitable mix of dataplanes

- Exploit heterogeneous resources.
- Meet resource & performance objectives.

Placement

- Analyze program’s use of resources.
- Split into set of 1-1 dataplane programs.
Current paradigm: dataplane program → one dataplane

Dataplane Disaggregation:

dataplane program → suitable mix of dataplanes

- Exploit heterogeneous resources.
- Meet resource & performance objectives.
- Exploit heterogeneous resources.
- Meet resource & performance objectives.

Placement

- Analyze program’s use of resources.
- Split into set of 1-1 dataplane programs.

- Hand-over control between dataplanes.
- Synchronize state.
- Detect and handle faults.
Current paradigm: dataplane program → one dataplane

Dataplane Disaggregation:
 dataplane program → suitable mix of dataplanes

- Analyze program’s use of resources.
- Split into set of 1-1 dataplane programs.
- Exploit heterogeneous resources.
- Meet resource & performance objectives.
- Hand-over control between dataplanes.
- Synchronize state.
- Detect and handle faults.

- No language or hardware changes.
- Changing programs, topology, and hardware.
- Ingest various data: program, topology, resource information, constraints, objectives.
- Scoping network programming and operation.
- Consider and explain multiple possible solutions.
- Multiple programs in same network.
- Diagnosis and debugging.
Flightplan

Dataplane Disaggregation:

- Analyze program’s use of resources.
- Split into set of 1-1 dataplane programs.

- Exploit heterogeneous resources.
- Meet resource & performance objectives.

- Hand-over control between dataplanes.
- Synchronize state.
- Detect and handle faults.

- No language or hardware changes.
- Changing programs, topology, and hardware.
- Ingest various data: program, topology, resource information, constraints, objectives.
- Scoping network programming and operation.
- Consider and explain multiple possible solutions.
- Multiple programs in same network.
- Diagnosis and debugging.

Placement

Code, tests, scripts, data, documentation:

https://flightplan.cis.upenn.edu/
Flightplan
Flightplan
Flightplan
Flightplan
Idea from symbolic AI: rule-based search.
Flightplan
Example program (Crosspod)

```plaintext

bit<1> compressed_link = 0;
bit<1> run_fec_egress = 0;
...
flyto(Compress);
// If heading out on a multiplexed link, then header compress.
egress_compression.apply(meta.egress_spec, compressed_link);
if (compressed_link == 1) {
    header_compress(forward);
    if (forward == 0) {
        drop();
        return;
    }
}
flyto(FEC_Encode);
check_run_FEC_egress.apply();
// If heading out on a lossy link, then FEC encode.
if (run_fec_egress == 1) {
    ...
    classification.apply(hdr, proto_and_port); // Sets hdr.fec.isValid()
    if (hdr.fec.isValid()) {
        encoder_params.apply(hdr.fec.traffic_class, k, h);
        update_fec_state(hdr.fec.traffic_class, k, h,
                         hdr.fec.block_index, hdr.fec.packet_index);
        hdr.fec.orig_ether_type = hdr.eth.type;
        FEC_ENCODE(hdr.fec, k, h);
    }
...  
```
Abstract program

(Rule generation is fully automatic)
Flightplan
Flightplan
Abstract Resource Semantics

<table>
<thead>
<tr>
<th>Component</th>
<th>Rate Condition</th>
<th>Rate Impact</th>
<th>Latency Impact</th>
<th>Latency Impact</th>
<th>Power Impact</th>
<th>Cost Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Rate $&lt; 2 \times 10^8$</td>
<td>Lat. $\mapsto$ Lat. $+ 7.4 \times 10^{-3}$</td>
<td>Rate $\mapsto$ Rate $\times \frac{189.9}{194.75}$</td>
<td>Power $\mapsto$ Power $+ 150$ W</td>
<td>Cost $\mapsto$ Cost $+ 5$</td>
<td></td>
</tr>
<tr>
<td>FPGA</td>
<td>Rate $&lt; 9.5 \times 10^9$</td>
<td>Lat. $\mapsto$ Lat. $+ 6.44 \times 10^{-6}$</td>
<td>Rate $\mapsto$ Rate $\times \frac{9.15}{9.3}$</td>
<td>$\langle$LUTs$\rangle$ $\mapsto$ LUTs $+ 24.4%$</td>
<td>$\langle$BRAMs$\rangle$ $\mapsto$ BRAMs $+ 54.4%$</td>
<td>FF $\mapsto$ FF $+ 15.8%$</td>
</tr>
</tbody>
</table>
Flightplan
Runtime: Fault Detection + Handling

Two mechanisms

- In-dataplane: +ve and -ve Acks.
- Strobes from control program.
Evaluation

- **Simulation:**
  - Scale of the network (featuring various programs)
  - Overhead
  - Disaggregation (different programs split in different ways)
  - Fail-over

- **Test-bed:**
  - Throughput, latency, power, resource utilization
  - Plan comparisons for hardware alternatives
  - Single-feature evaluation
Fig 10: Resources & Behavior vs Functions (Testbed experiment)
Fig 9: Plan comparison (Testbed + Simulation)

Reciprocal of throughput rate

Max. Performance
Legacy Extender
Server Offload (Tofino)
Server Offload (Arista)

Inner is better
Fig 7: Multiple Programs vs Runtimes vs Splits
in same network (Simulation)
Fig 7: Multiple Programs vs Runtimes vs Splits in same network (Simulation)
Fig 7: Multiple Programs vs Runtimes vs Splits in same network (Simulation)
**FLIGHTPLAN DEMO**

Choose an Experiment...

- Start
- About

**MSc students:** Heena Nagda (GATech), Rakesh Nagda (Penn)

Other features: graphs, multimedia cues (e.g., icons, packet structure), ...

https://flightplan.cis.upenn.edu/
Flightplan: Dataplane Disaggregation and Placement for P4 programs

Nik Sultana  John Sonchack  Hans Giesen  Isaac Pedisich  Zhaoyang Han
Nishanth Shyamkumar  Shivani Burad  André DeHon  Boon Thau Loo

University of Pennsylvania

https://flightplan.cis.upenn.edu/

NSDI’21