Flightplan: Dataplane Disaggregation and Placement for P4 programs

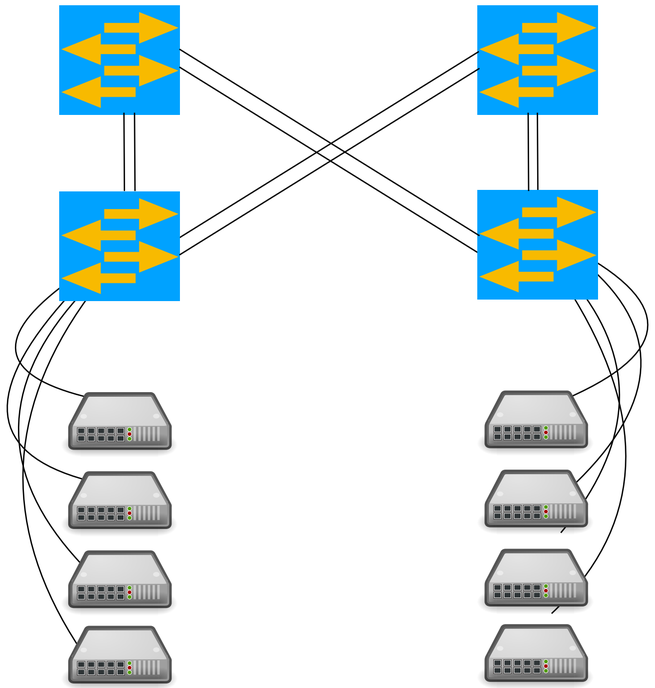
Nik Sultana John Sonchack Hans Giesen Isaac Pedisich Zhaoyang Han
Nishanth Shyamkumar Shivani Burad André DeHon Boon Thau Loo

University of Pennsylvania

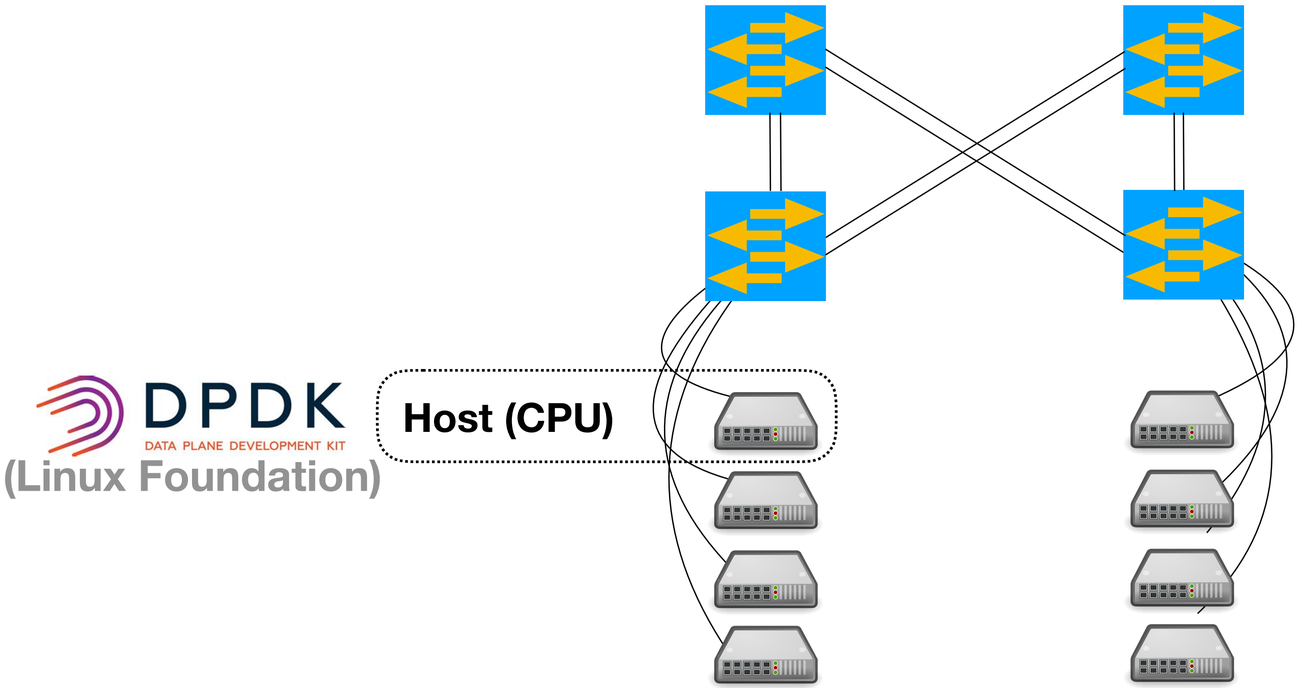
https://flightplan.cis.upenn.edu/

NSDI'21

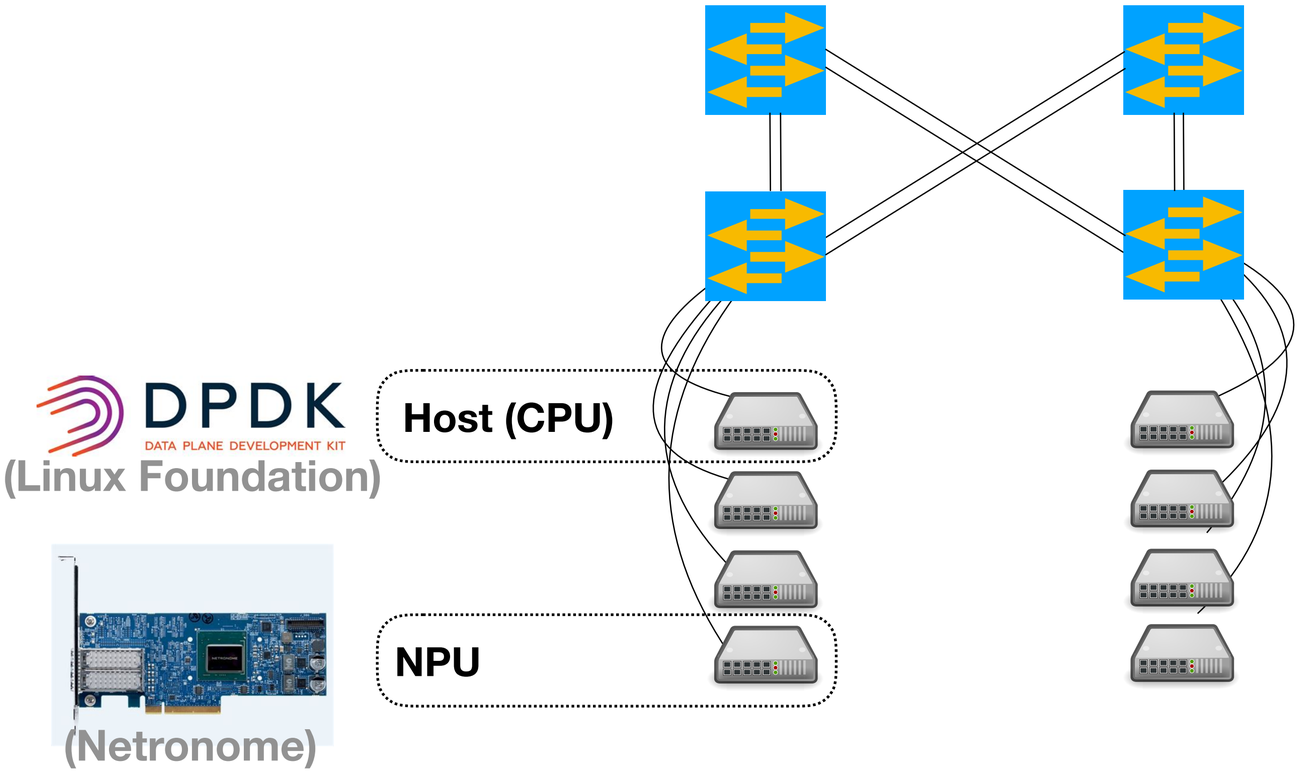
Dataplane Programmability



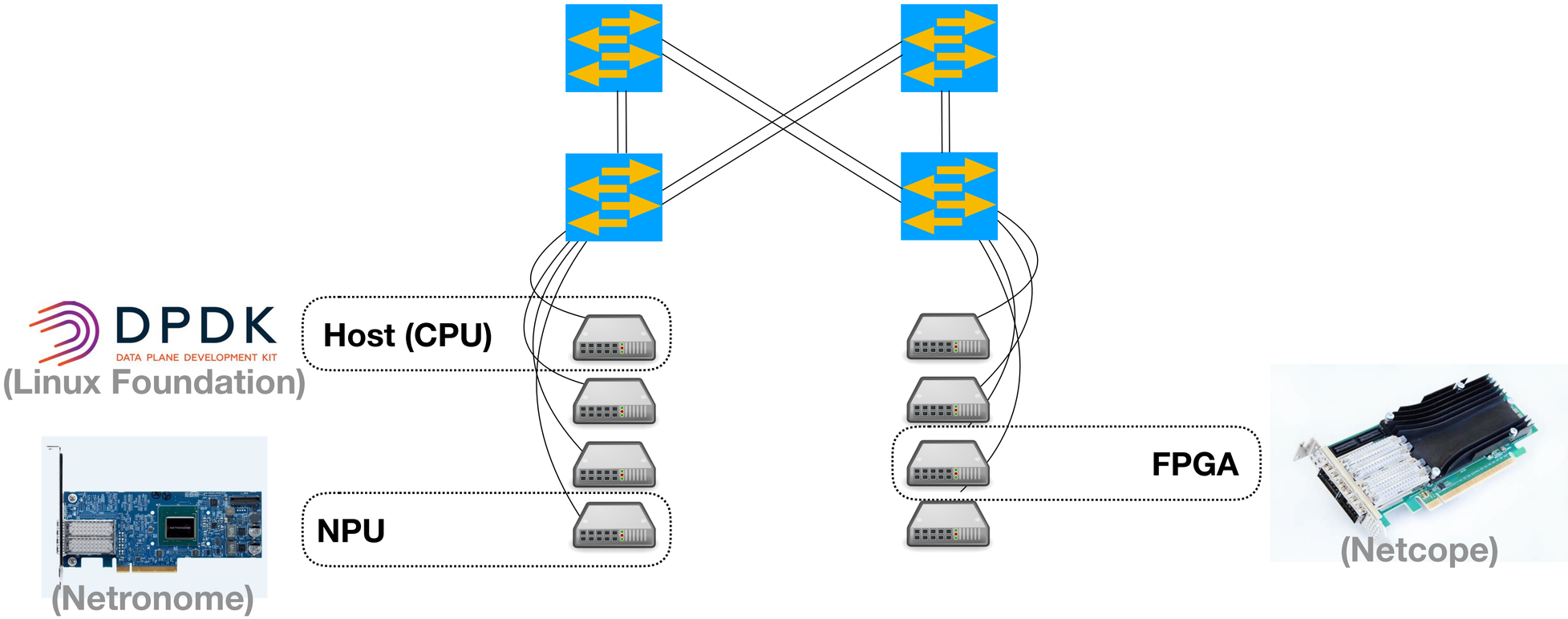
Dataplane Programmability



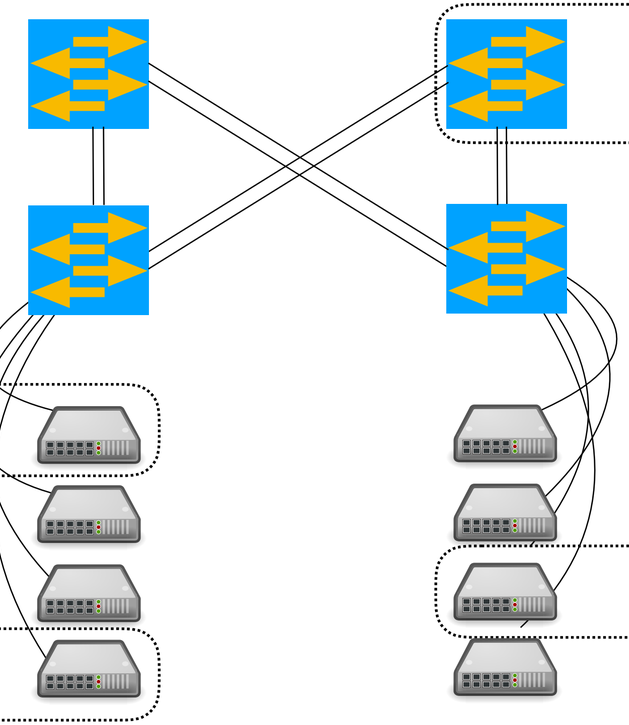
Dataplane Programmability



Dataplane Programmability



Dataplane Programmability



Switch

(P4 Language Consortium)

DPDK
DATA PLANE DEVELOPMENT KIT

(Linux Foundation)

Host (CPU)

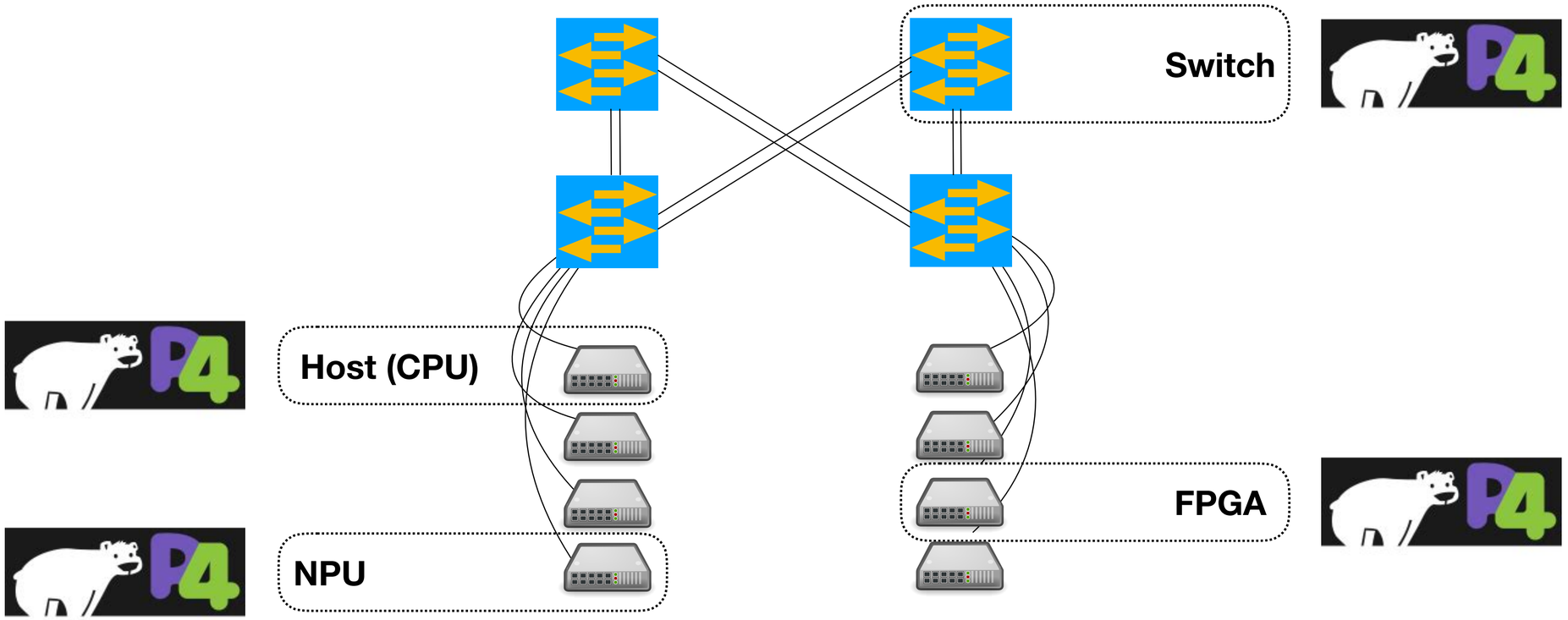
NPU

FPGA

(Netronome)

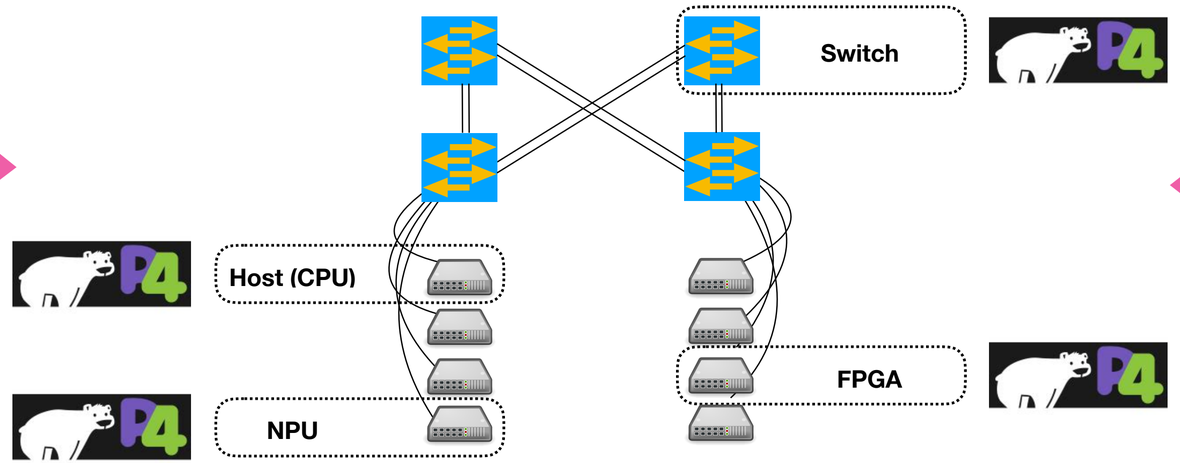
(Netcope)

Dataplane Programmability



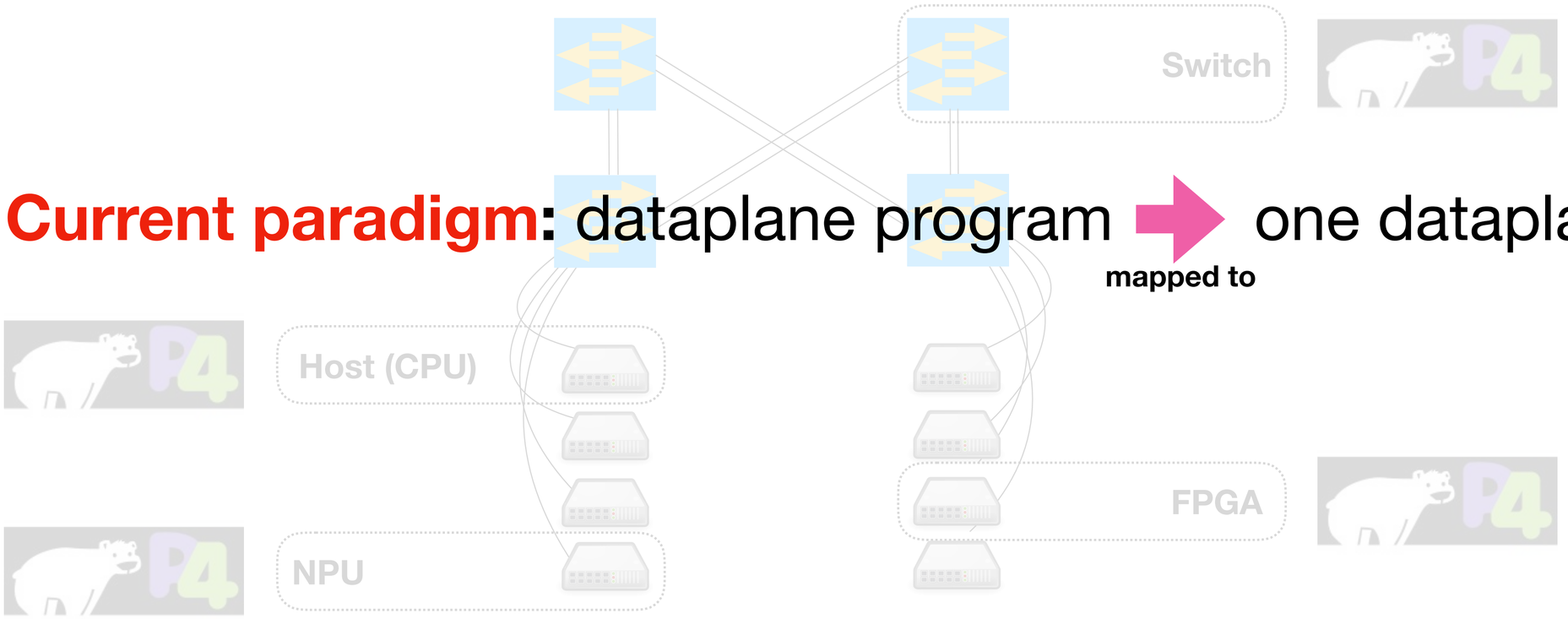
Dataplane Programmability

Research



Products

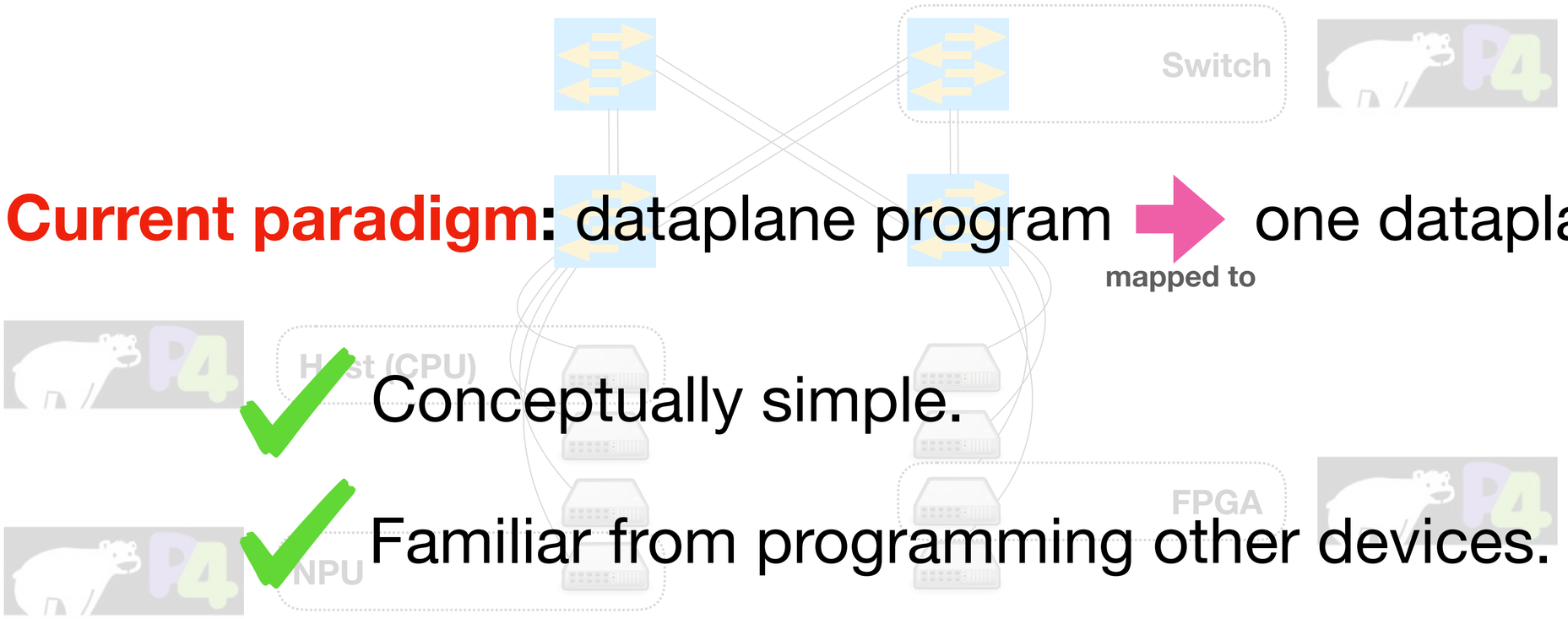
Dataplane Programmability



Current paradigm: dataplane program → one dataplane (device)

mapped to

Dataplane Programmability



Current paradigm: dataplane program → one dataplane (device)

mapped to

✓ Conceptually simple.

✓ Familiar from programming other devices.

✗ Mismatch with Software-Defined Networking (the overarching vision).

Current paradigm: dataplane program → one dataplane

mapped to

xor

xor

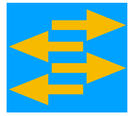
xor

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

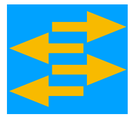
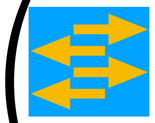
mapped to


xor


xor

xor


New paradigm:
dataplane program → suitable mix of dataplanes

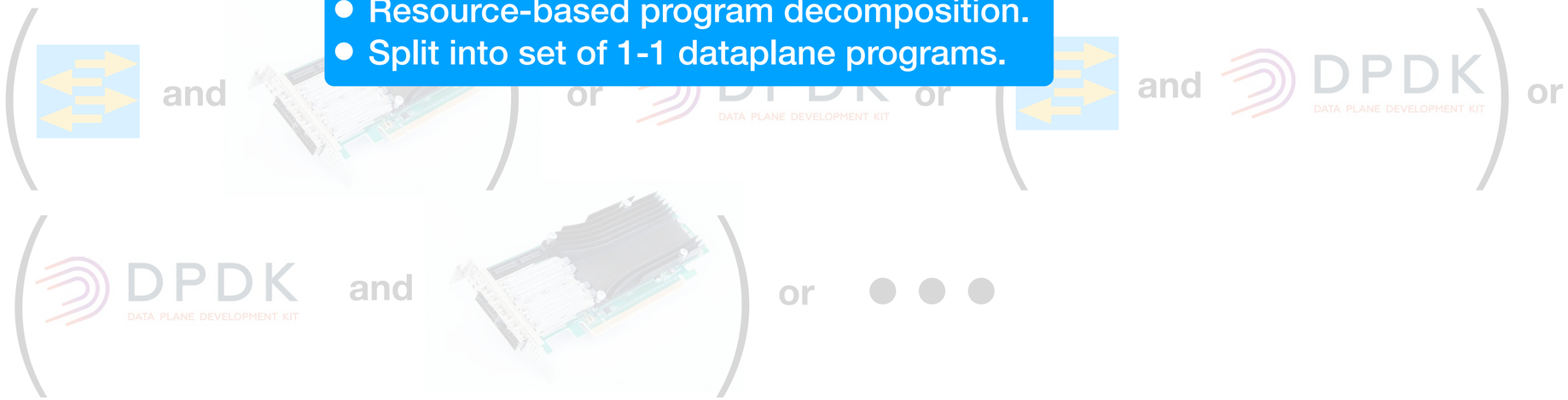
( and ) or  or ( and ) or

( and ) or ● ● ●

Current paradigm: dataplane program → one dataplane

New paradigm:
dataplane program → suitable mix of dataplanes

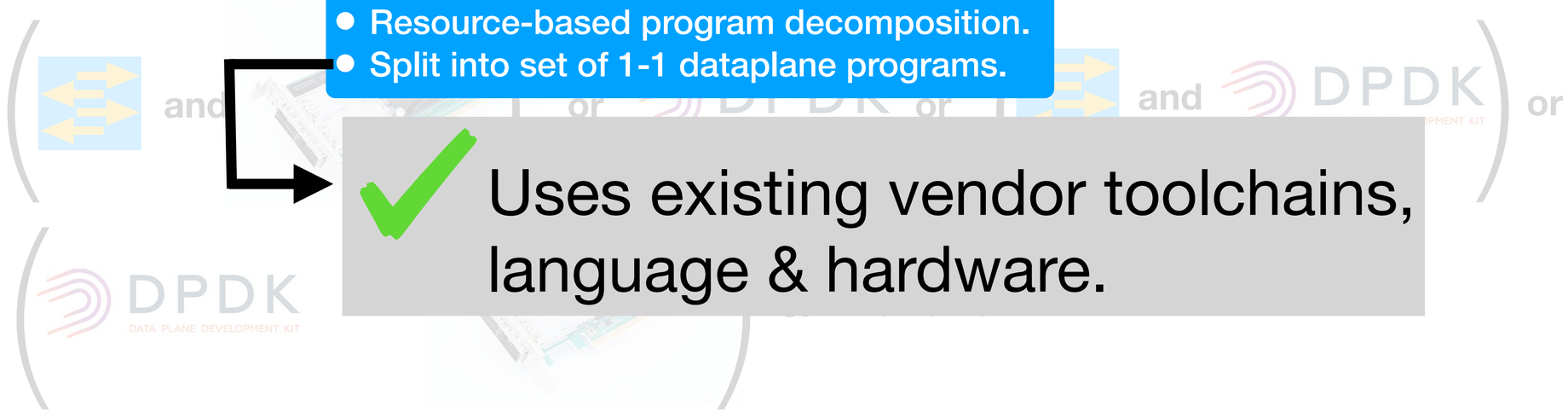
- Resource-based program decomposition.
- Split into set of 1-1 dataplane programs.



Current paradigm: dataplane program → one dataplane

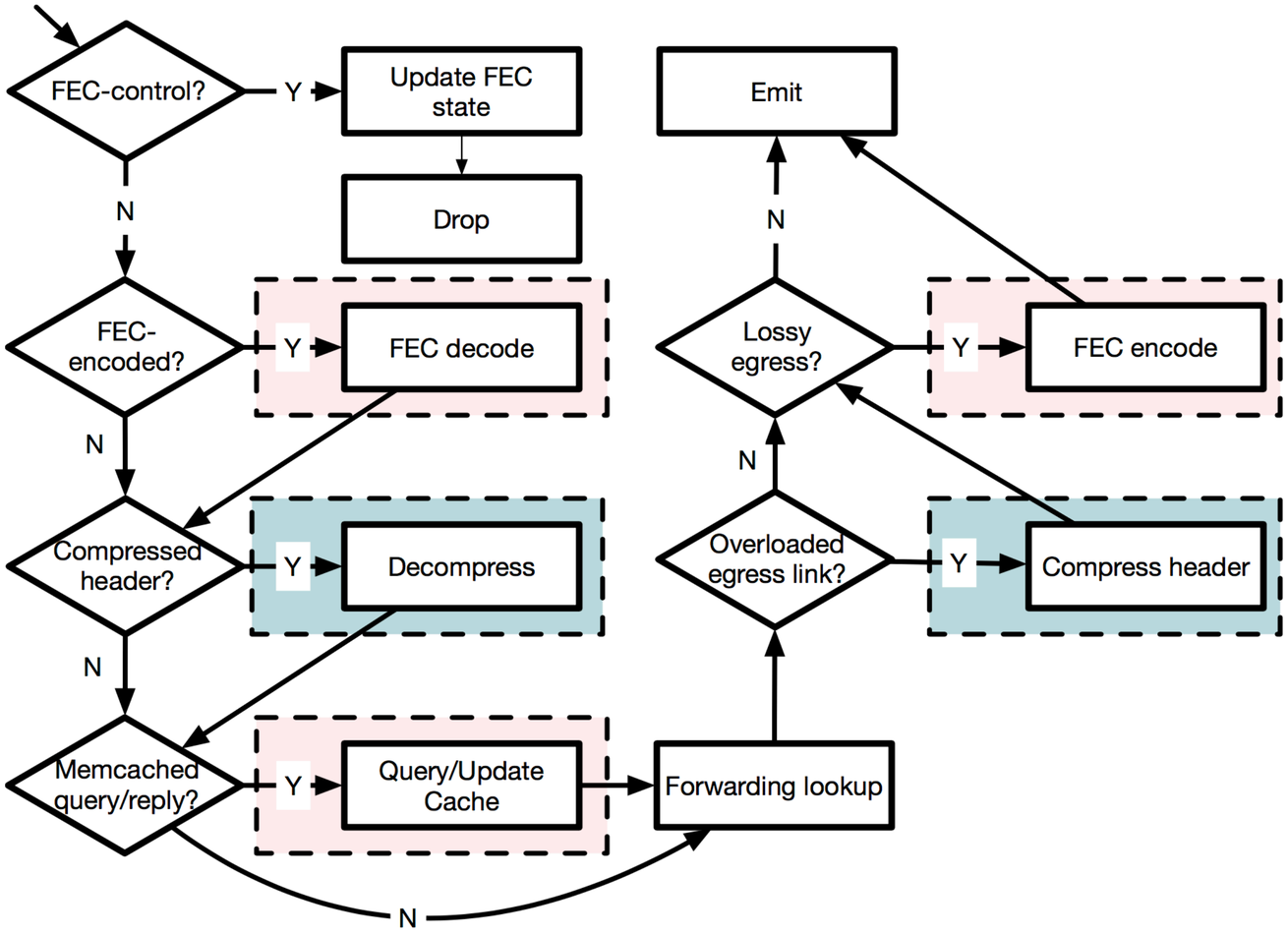
New paradigm:
dataplane program → suitable mix of dataplanes

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

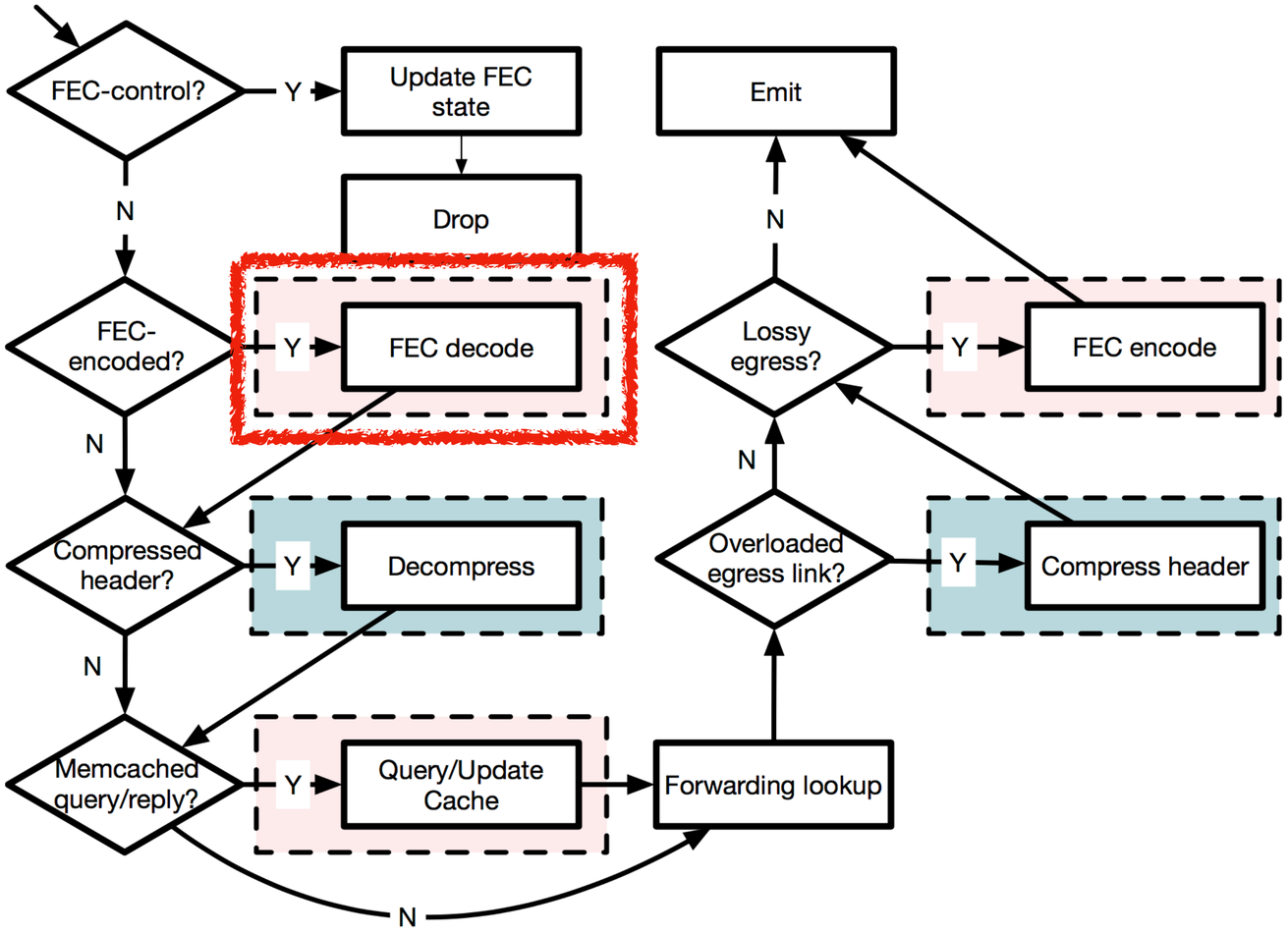


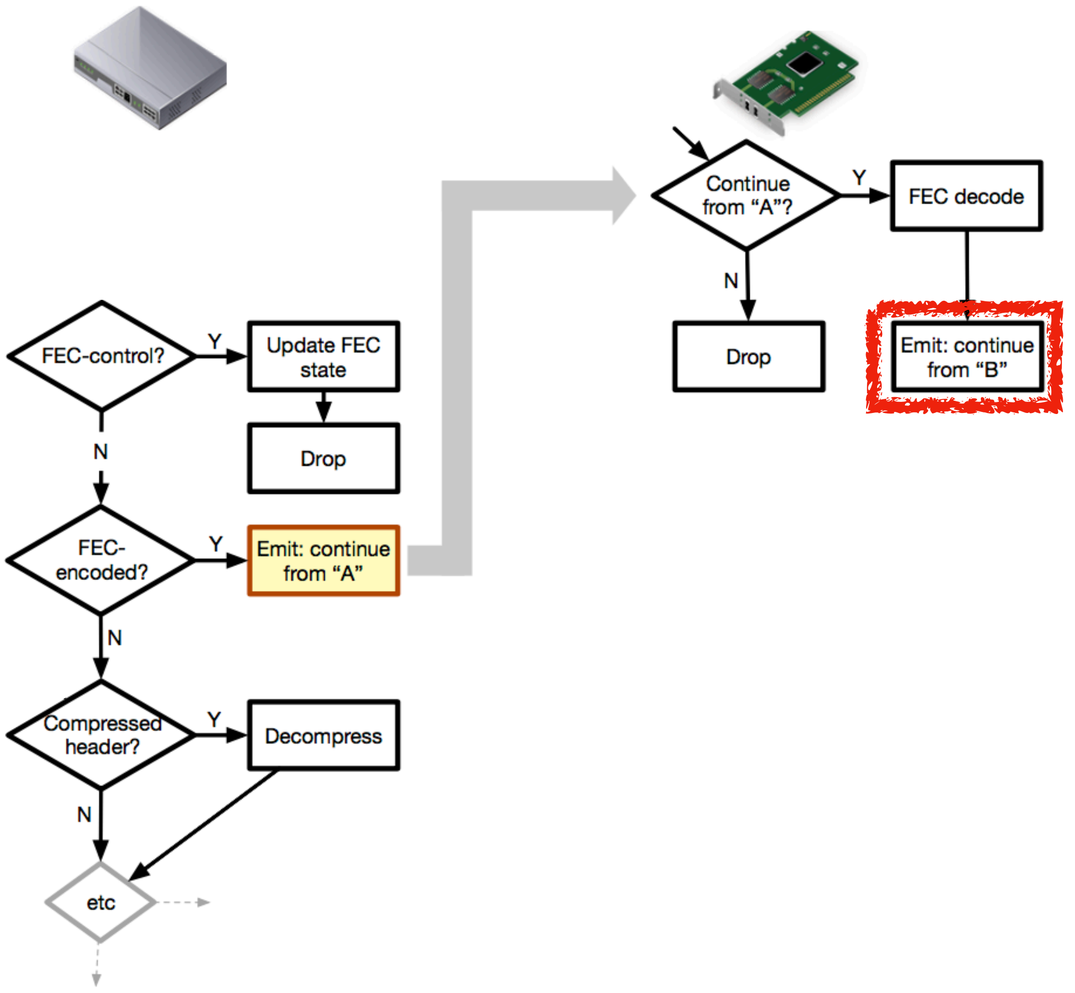
✓ Uses existing vendor toolchains, language & hardware.

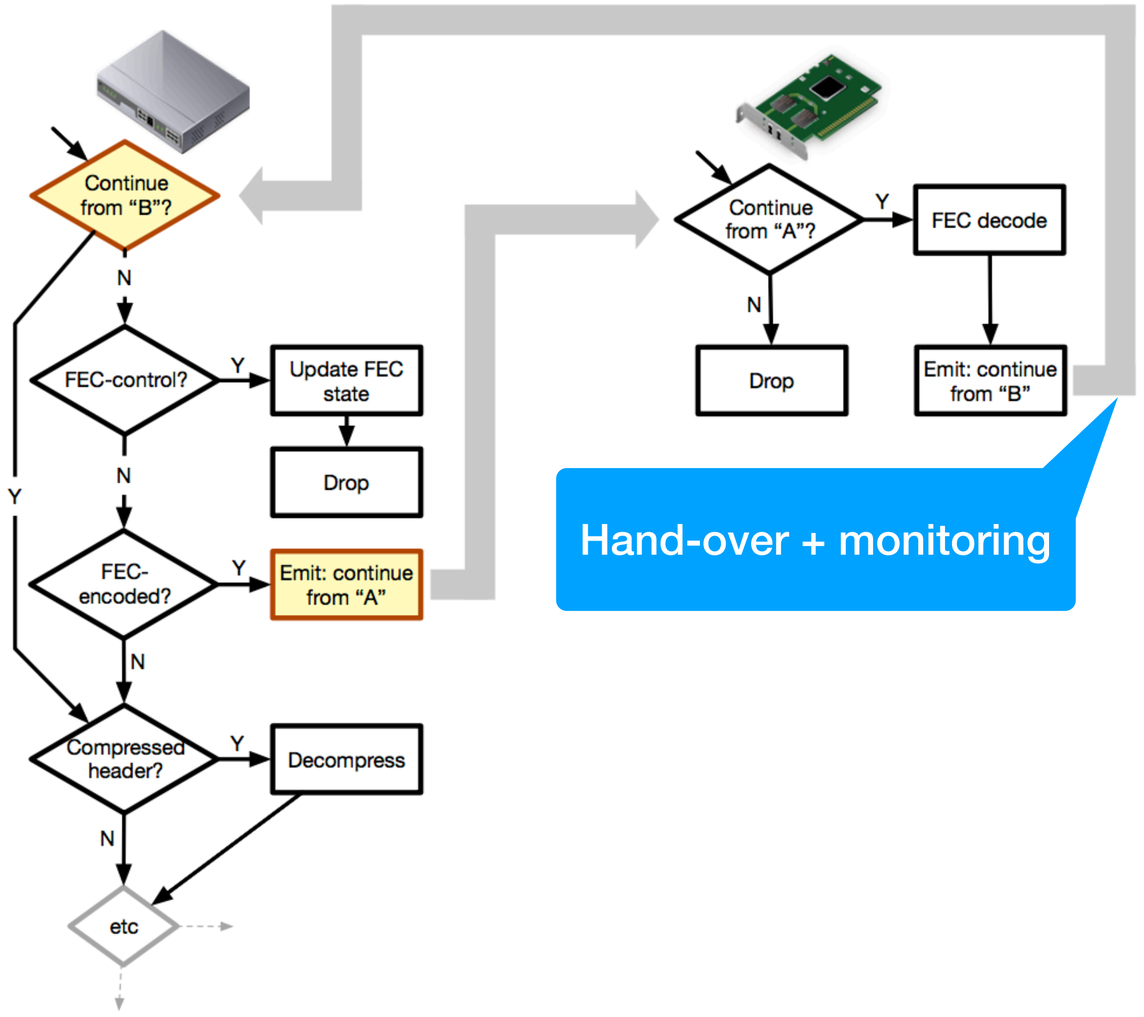
Example: “Crosspod” Program

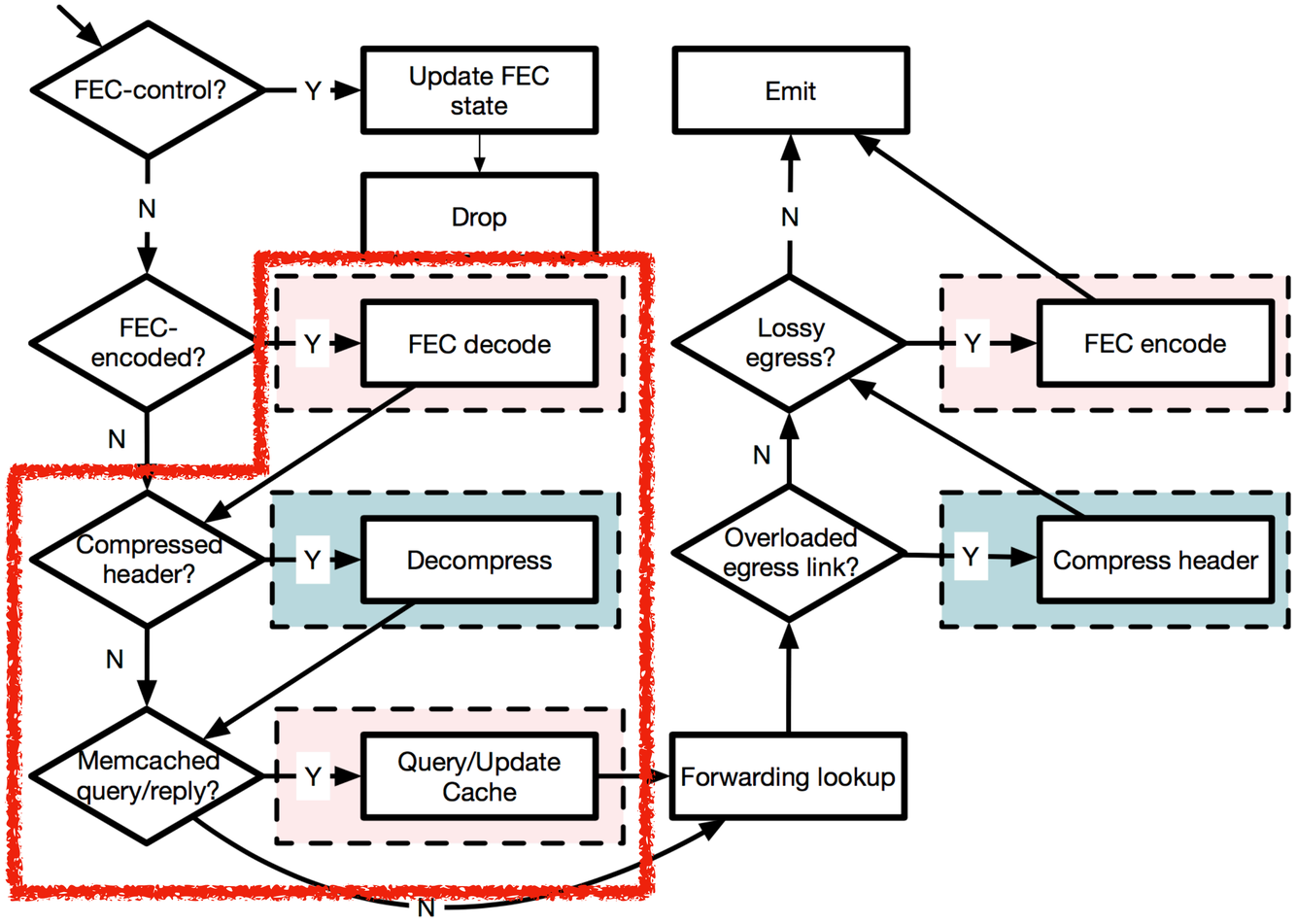


Example: “Crosspod” Program

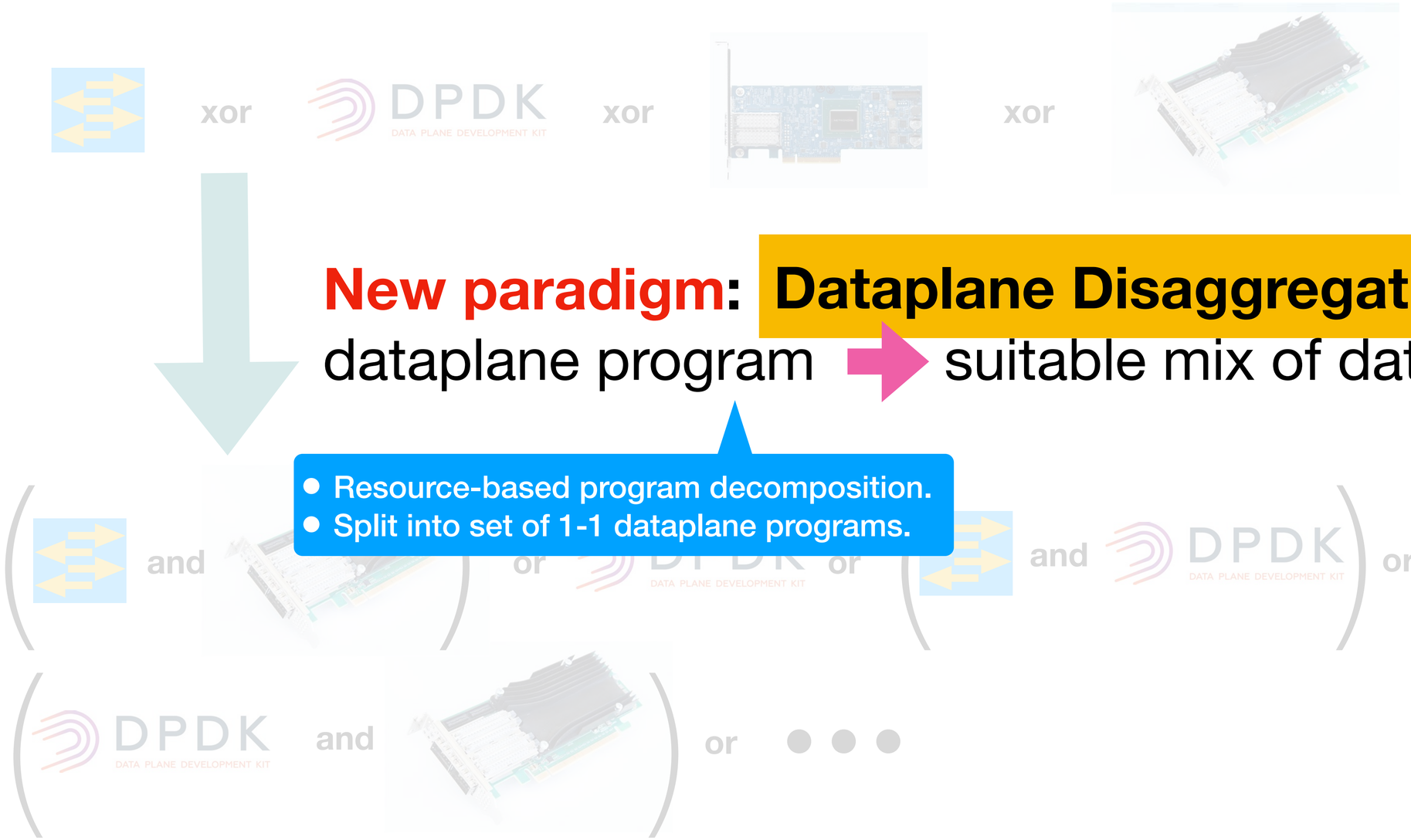



FEC-control?
Y
Update FEC state
Drop
N
FEC-encoded?
Y
Emit: continue from “A”
N
Compressed header?
Y
Decompress
N
etc
Continue from “A”?
Y
FEC decode
N
Drop
Emit: continue from “B”


Continue from “B”?
N
FEC-control?
Y
Update FEC state
Drop
N
Y
FEC-encoded?
Y
Emit: continue from “A”
N
Compressed header?
Y
Decompress
N
etc
Continue from “A”?
Y
FEC decode
N
Drop
Emit: continue from “B”
Hand-over + monitoring


FEC-control?
Y
Update FEC state
Drop
N
FEC-encoded?
Y
FEC decode
N
Compressed header?
Y
Decompress
N
Memcached query/reply?
Y
Query/Update Cache
N
Forwarding lookup
Overloaded egress link?
Y
Compress header
N
Lossy egress?
Y
FEC encode
N
Emit

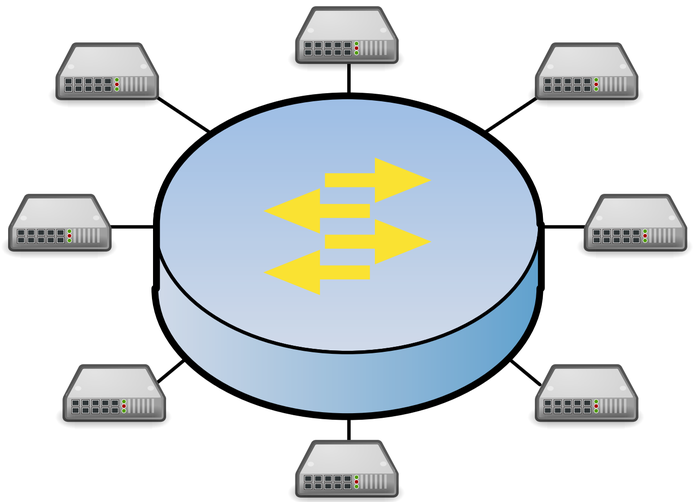
Current paradigm: dataplane program → one dataplane



Dataplane Disaggregation

(≠ Server Disaggregation)

Virtual Dataplane ➔ Set of Physical Dataplanes



“One big switch” ➔ “One big *programmable* switch”

Current paradigm: dataplane program → one dataplane

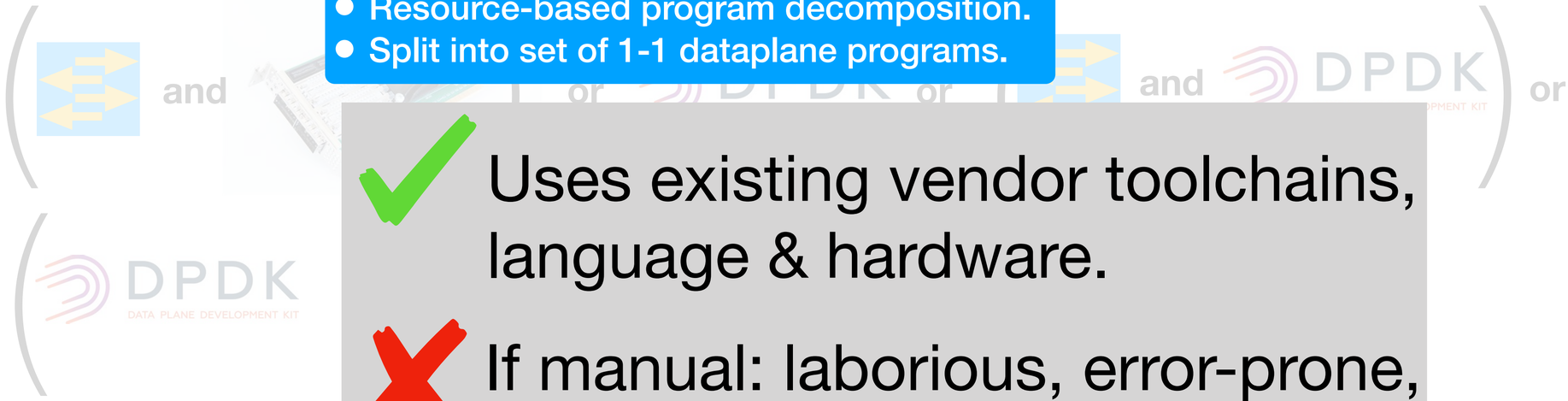
How to implement?

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.

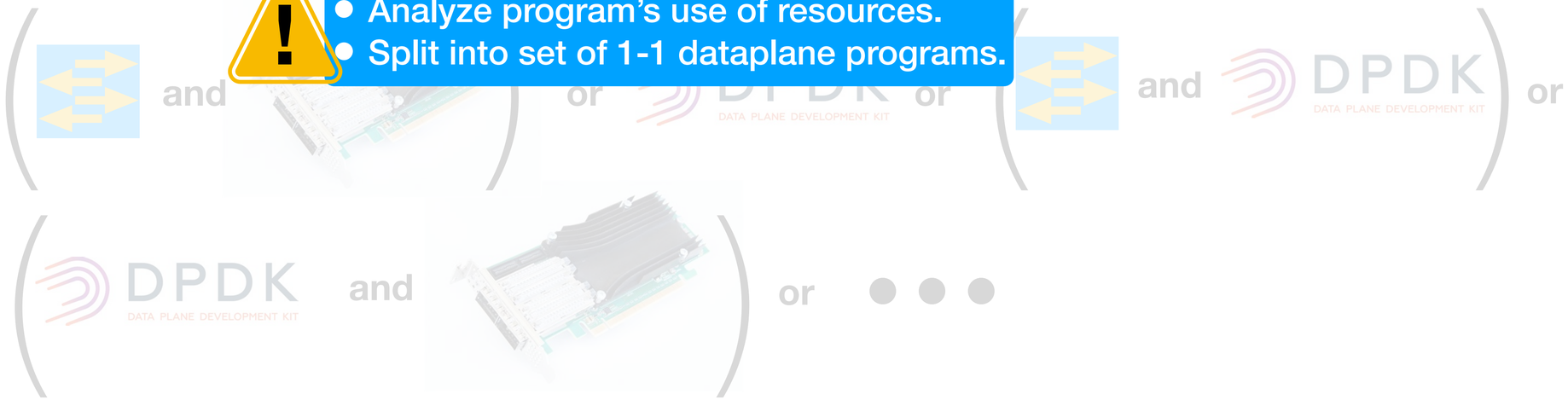
✗ If manual: laborious, error-prone, hard to change.

Current paradigm: dataplane program → one dataplane

Dataplane Disaggregation:

dataplane program → suitable mix of dataplanes

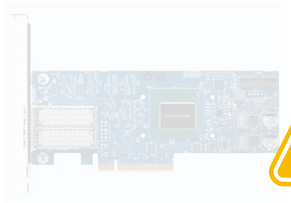
- Analyze program's use of resources.
- Split into set of 1-1 dataplane programs.



Current paradigm: dataplane program → one dataplane

xor

xor



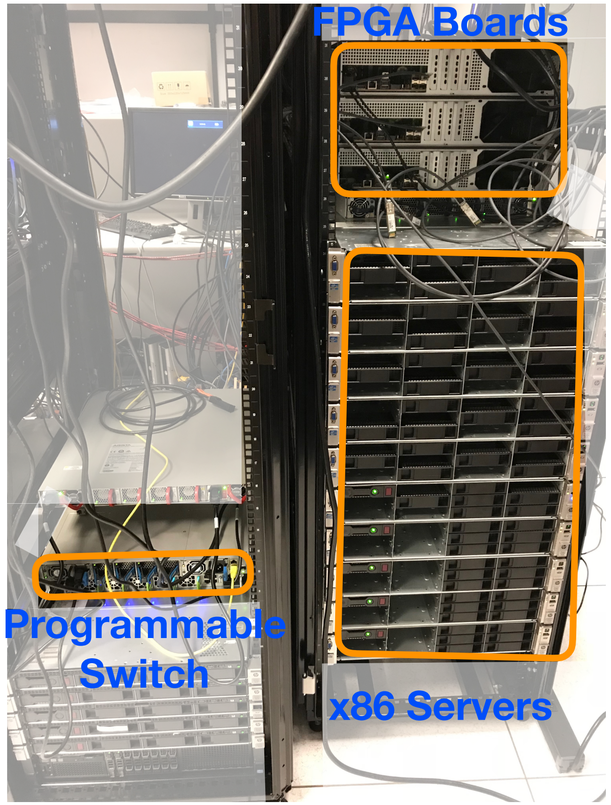
- Exploit heterogeneous resources.
- Meet resource & performance objectives.

Dataplane Disaggregation:

dataplane program → suitable mix of dataplanes

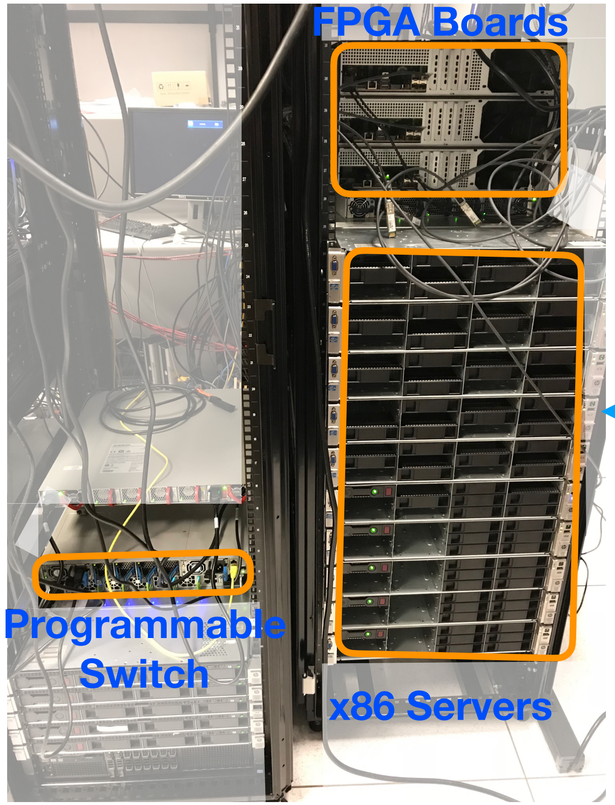
- Analyze program's use of resources.
- Split into set of 1-1 dataplane programs.

Heterogeneity



Testbed Deployment

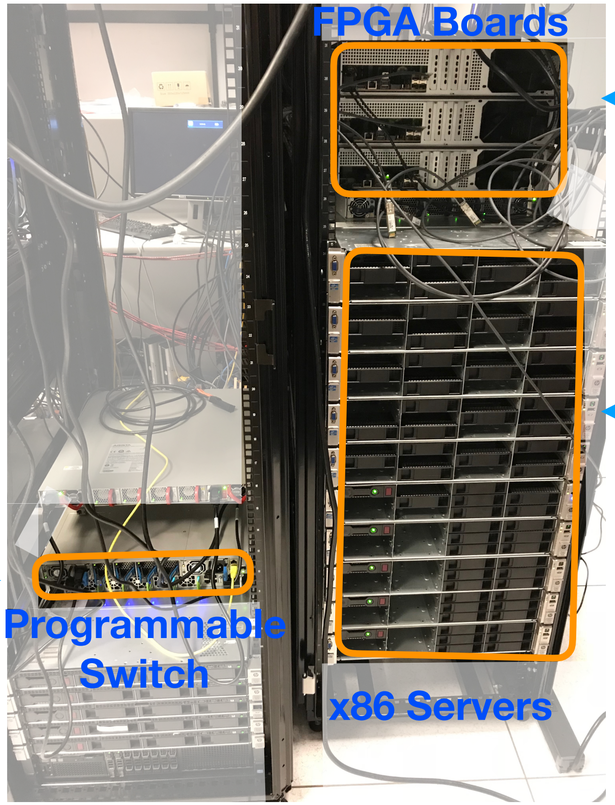
Heterogeneity



Testbed Deployment

~110W	Power
~$5K	Cost
~90Gbps	Throughput

Heterogeneity



Testbed Deployment

Current paradigm: dataplane program → one dataplane

xor

xor

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

Dataplane Disaggregation:

dataplane program → suitable mix of dataplanes

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

and

or

or

and

or

DPDK
DATA PLANE DEVELOPMENT KIT

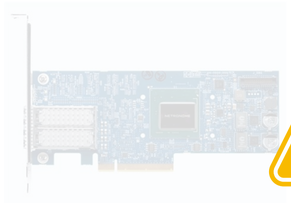
and

or ● ● ●

Current paradigm: dataplane program → one dataplane

xor

xor



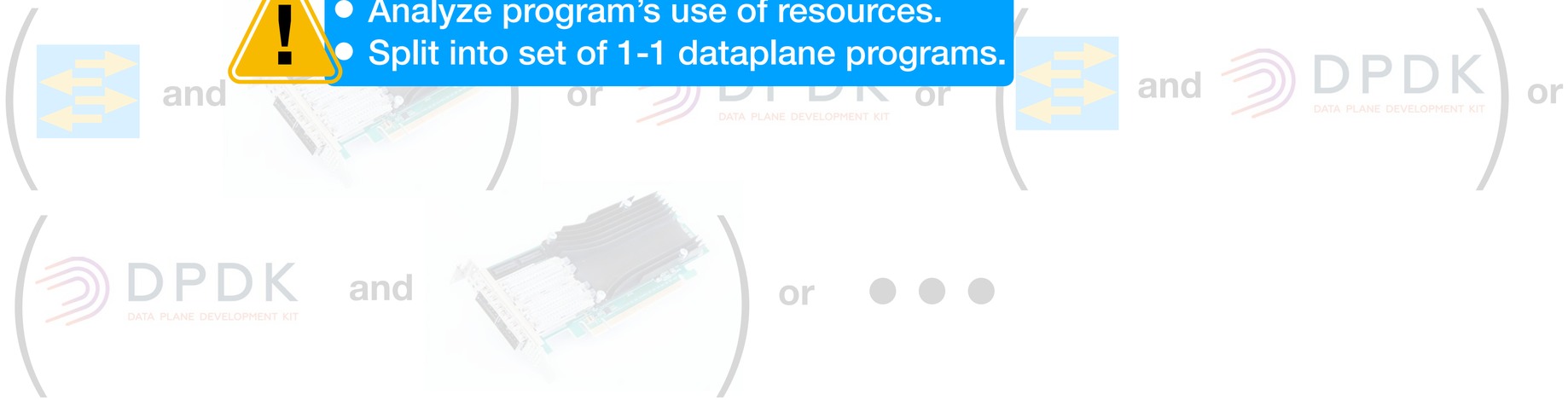
- Exploit heterogeneous resources.
- Meet resource & performance objectives.

Dataplane Disaggregation:

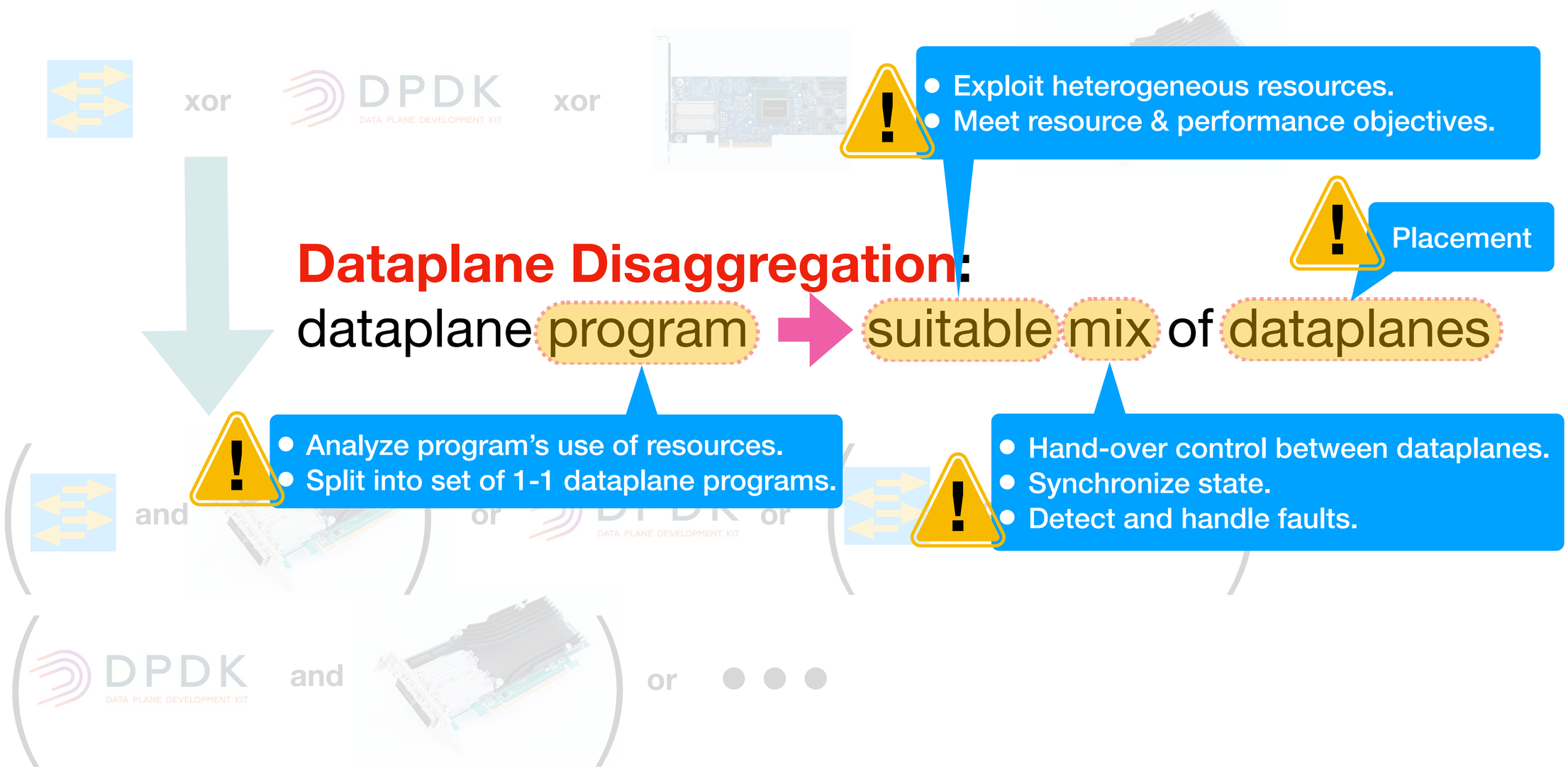
dataplane program → suitable mix of dataplanes

Placement

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



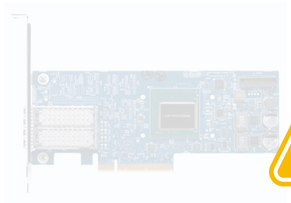
Current paradigm: dataplane program → one dataplane



Current paradigm: dataplane program → one dataplane

xor

xor



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

Dataplane Disaggregation:

dataplane program → suitable mix of dataplanes

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.

and

or

or

and

- 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.

Code, tests, scripts, data, documentation:
https://flightplan.cis.upenn.edu/

Flightplan

Dataplane Disaggregation:

dataplane program → suitable mix of dataplanes

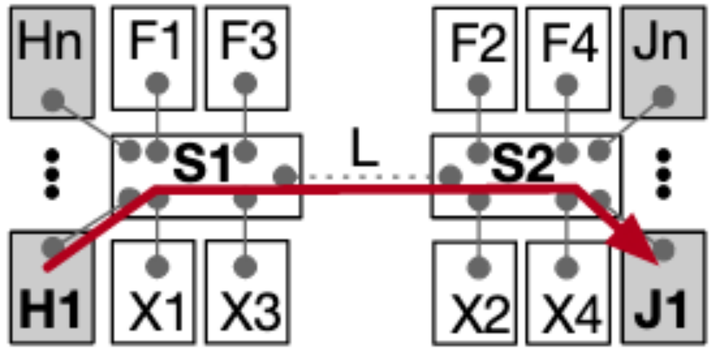
- Exploit heterogeneous resources.
- Meet resource & performance objectives.

Placement

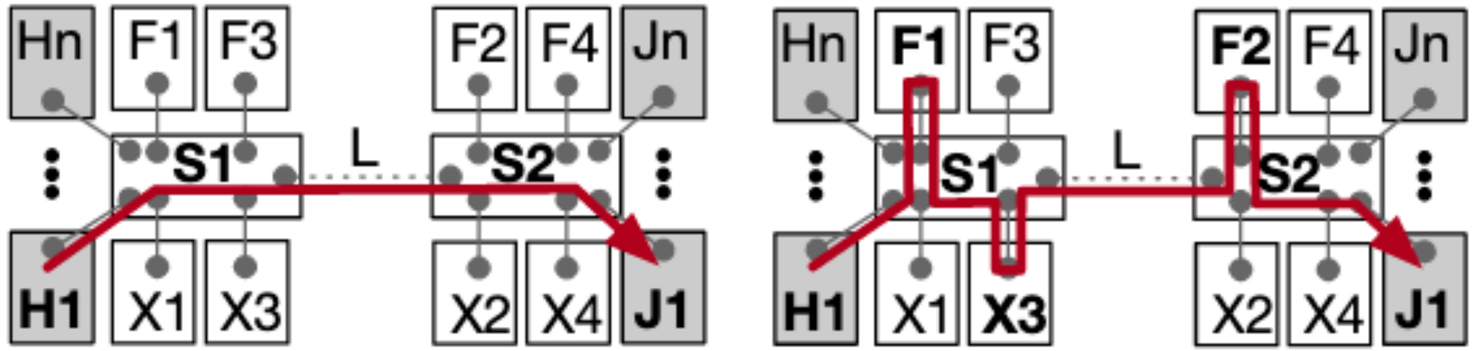
- Analyze program's use of resources.
- Split into set of 1-1 dataplane programs.

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

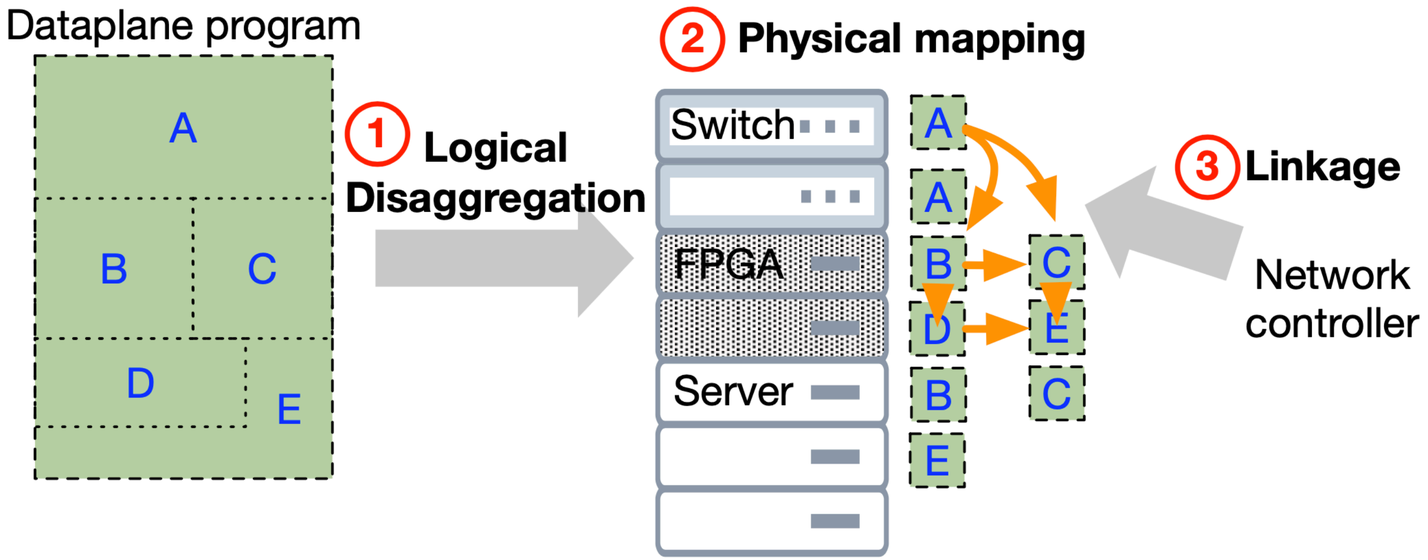
- No language or hardware changes.
- Changing programs, topology, and hardware.
- Ingest various data: program, topology, resource information, constraints, objectives.
- Scoping network programming and operation.
- Consider and explain multiple possible solutions.
- Multiple programs in same network.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



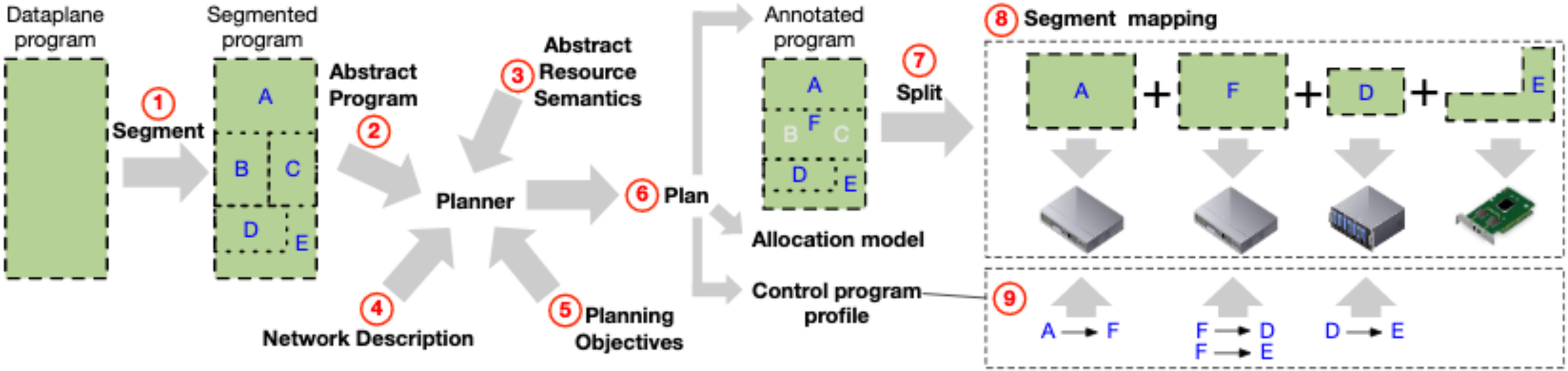
Flightplan


Hn
F1
F3
F2
F4
Jn
S1
L
S2
H1
X1
X3
X2
X4
J1
Hn
F1
F3
F2
F4
Jn
S1
L
S2
H1
X1
X3
X2
X4
J1

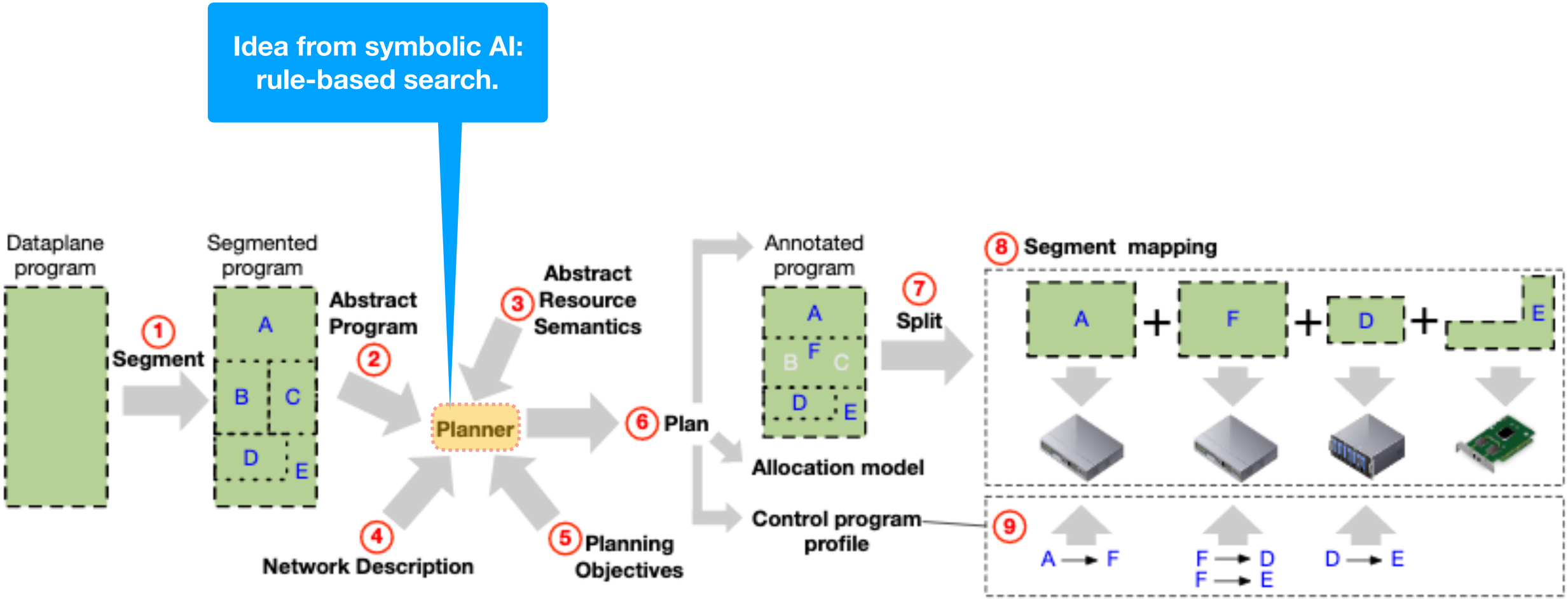
Flightplan


Dataplane program
A
B
C
D
E
1 Logical Disaggregation
2 Physical mapping
Switch
FPGA
Server
A
A
B
C
D
E
B
C
E
3 Linkage
Network controller

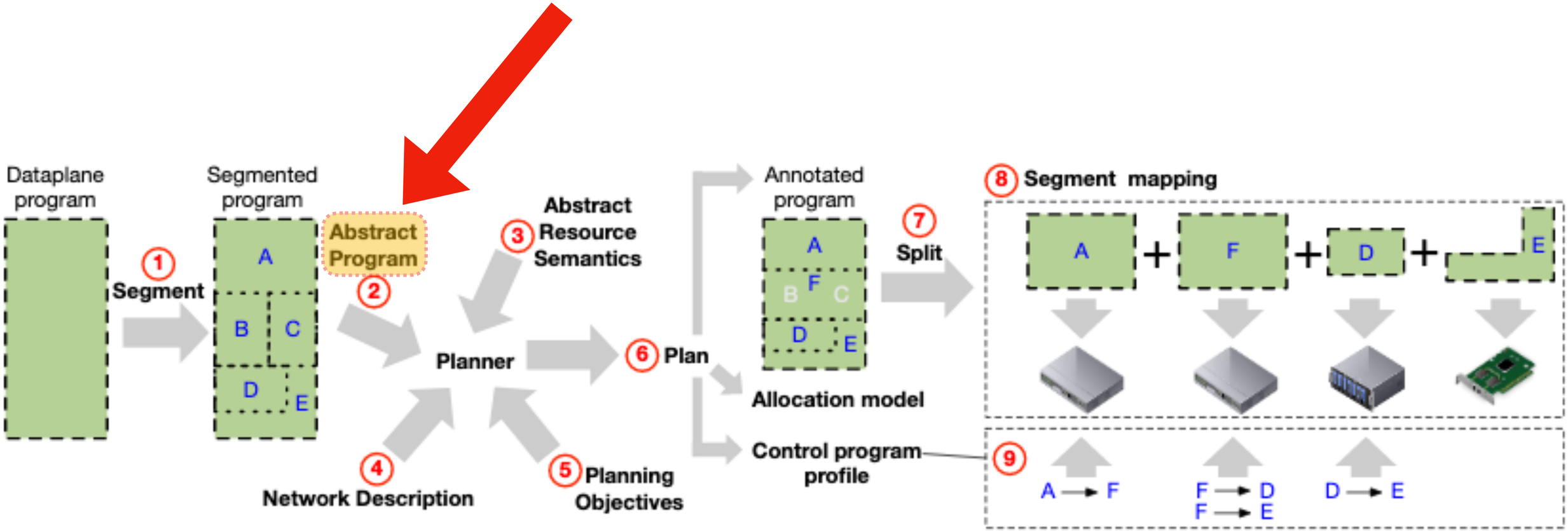
Flightplan


Dataplane program
1 Segment
Segmented program
A
B C
D E
Abstract Program
2
3 Abstract Resource Semantics
Planner
4 Network Description
5 Planning Objectives
6 Plan
Annotated program
A
B F C
D E
Allocation model
Control program profile
7 Split
8 Segment mapping
A + F + D + E
9
A → F
F → D
F → E
D → E

Flightplan


Idea from symbolic AI: rule-based search.
Dataplane program
1 Segment
Segmented program
A
B C
D E
Abstract Program
2
3 Abstract Resource Semantics
Planner
4 Network Description
5 Planning Objectives
6 Plan
Annotated program
A
B F C
D E
7 Split
Allocation model
Control program profile
8 Segment mapping
A + F + D + E
9
A → F
F → D
F → E
D → E

Flightplan


Dataplane program
Segmented program
Abstract Program
1 Segment
2
3 Abstract Resource Semantics
Planner
4 Network Description
5 Planning Objectives
6 Plan
Annotated program
Allocation model
Control program profile
7 Split
8 Segment mapping
9
A
B
C
D
E
F
A → F
F → D
F → E
D → E

Flightplan

Example program (Crosspod)

Segment annotation

Resource dependence

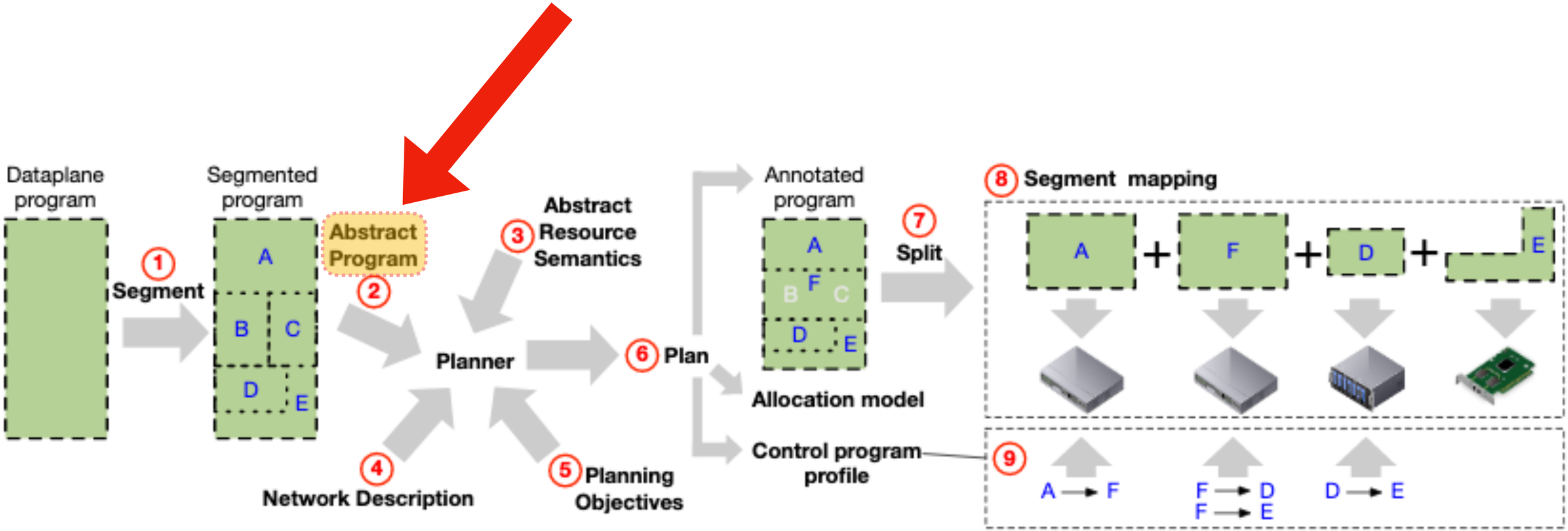
```
1   bit<1> compressed_link = 0;
2   bit<1> run_fec_egress = 0;
3   ...
4   flyto(Compress);
5   // If heading out on a multiplexed link, then header compress.
6   egress_compression.apply(meta.egress_spec, compressed_link);
7   if (compressed_link == 1) {
8       header_compress(forward);
9       if (forward == 0) {
10          drop();
11          return;
12      }
13  }
14  flyto(FEC_Encode);
15  check_run_FEC_egress.apply();
16  // If heading out on a lossy link, then FEC encode.
17  if (run_fec_egress == 1) {
18      ...
19      classification.apply(hdr, proto_and_port); // Sets hdr.fec.isValid()
20      if (hdr.fec.isValid()) {
21          encoder_params.apply(hdr.fec.traffic_class, k, h);
22          update_fec_state(hdr.fec.traffic_class, k, h,
23                           hdr.fec.block_index, hdr.fec.packet_index);
24          hdr.fec.orig_ethertype = hdr.eth.type;
25          FEC_ENCODE(hdr.fec, k, h);
26          ...
```

Abstract program

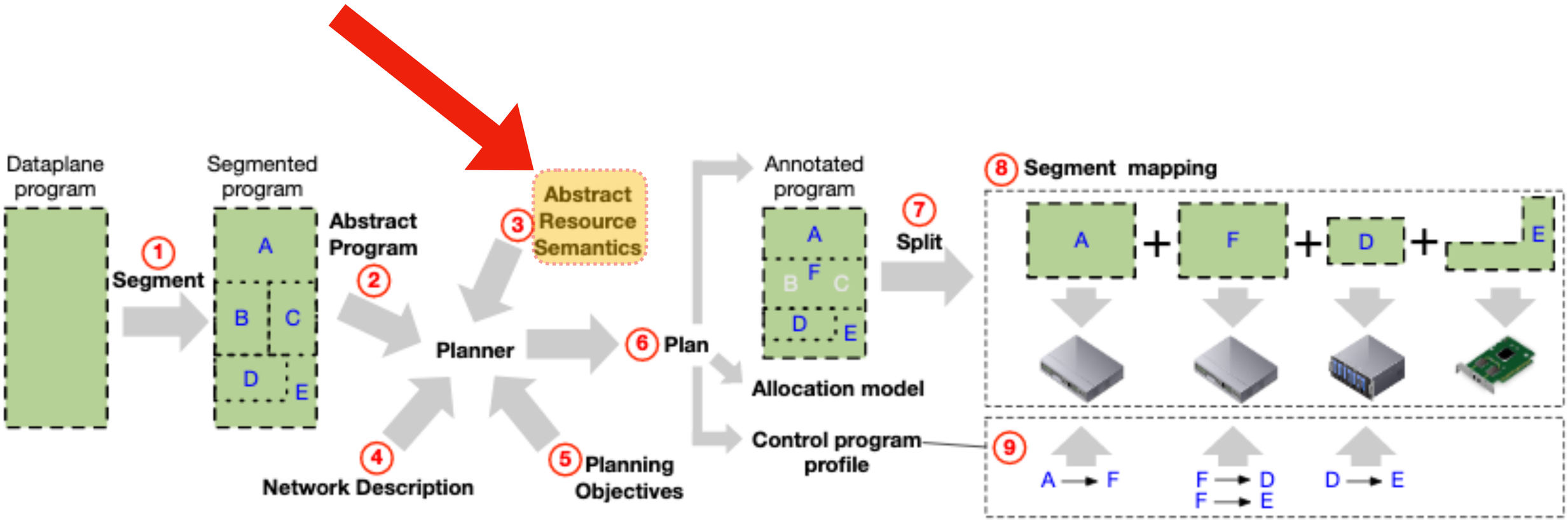
```
flyto(Compress);
// If heading out on a multiplexed link, then header compress.
egress_compression.apply(meta.egress_spec, compressed_link);
if (compressed_link == 1) {
    header_compress(forward);
    if (forward == 0) {
        drop();
        return;
    }
}
flyto(FEC_Encode);
```

(Rule generation is fully automatic)

$$\text{R2}\ \frac{\text{egress\_compres.} \qquad \text{header\_compress}}{\text{Compress}} \qquad \text{drop}$$


Dataplane program
Segmented program
Abstract Program
1 Segment
2
3 Abstract Resource Semantics
Planner
4 Network Description
5 Planning Objectives
6 Plan
Annotated program
Allocation model
Control program profile
7 Split
8 Segment mapping
9
A
B
C
D
E
F
A → F
F → D
F → E
D → E

Flightplan

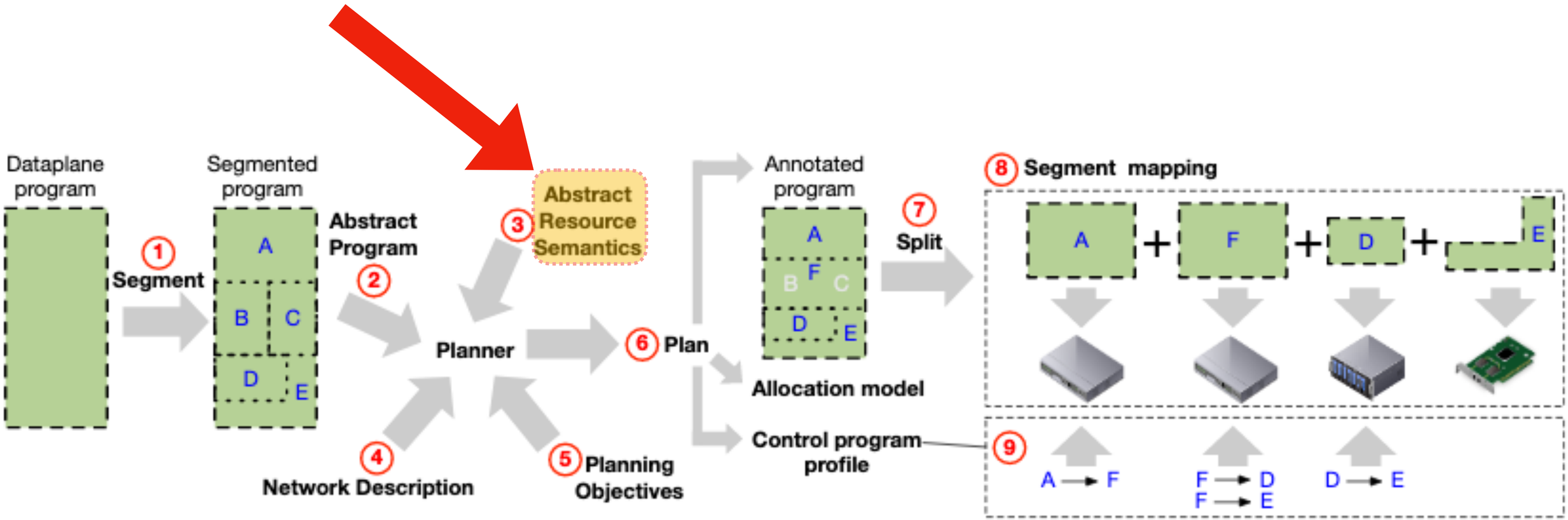

Dataplane program
1 Segment
Segmented program
A
B C
D E
Abstract Program
2
3
Abstract Resource Semantics
Planner
4
Network Description
5 Planning Objectives
6 Plan
Annotated program
A
B F C
D E
Allocation model
Control program profile
7 Split
8 Segment mapping
A + F + D + E
9
A → F
F → D
F → E
D → E

Flightplan

Abstract Resource Semantics

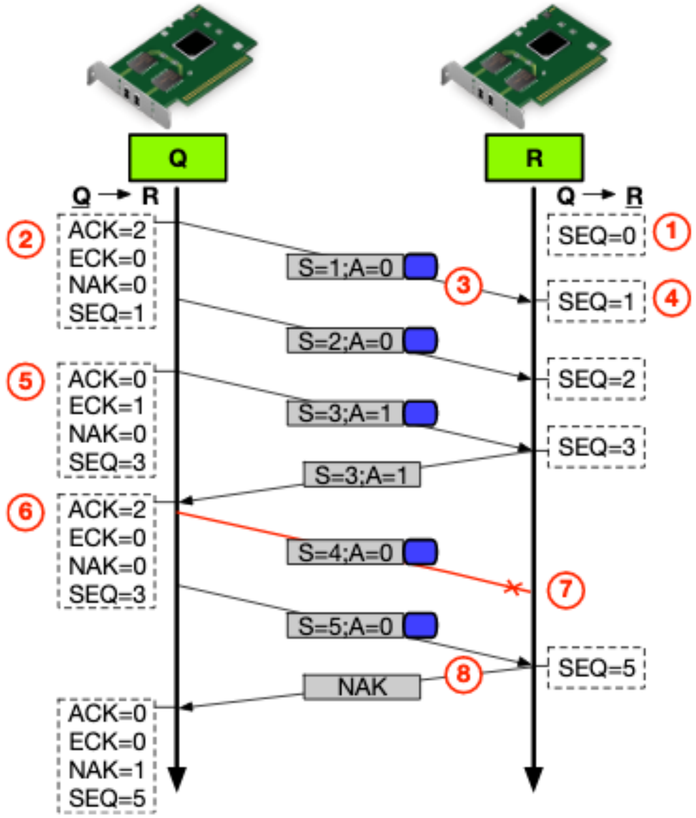
$$\frac{\begin{array}{c}\text{CPU} \quad \text{Rate} < 2 \times 10^8 \\ \text{PacketSize} > 1000\end{array}}{\text{header\_compress}} \left[\begin{array}{l} \text{Lat.} \mapsto \text{Lat.} + 7.4 \times 10^{-3} \\ \text{Rate} \mapsto \text{Rate} \times \frac{189.9}{194.75} \\ \text{once Power} \mapsto \text{Power} + 150\,\text{W} \\ \text{once Cost} \mapsto \text{Cost} + 5 \end{array}\right]$$

$$\frac{\begin{array}{c}\text{FPGA} \quad \text{Rate} < 9.5 \times 10^9 \\ \text{PacketSize} > 100\end{array}}{\text{header\_compress}} \left[\begin{array}{l} \text{Lat.} \mapsto \text{Lat.} + 6.44 \times 10^{-6} \\ \text{Rate} \mapsto \text{Rate} \times \frac{9.15}{9.3} \\ \langle \text{LUTs} \rangle \mapsto \text{LUTs} + 24.4\% \\ \langle \text{BRAMs} \rangle \mapsto \text{BRAMs} + 54.4\% \\ \langle \text{FF} \rangle \mapsto \text{FF} + 15.8\% \\ \text{once Power} \mapsto \text{Power} + 30\,\text{W} \\ \text{once Cost} \mapsto \text{Cost} + 2 \end{array}\right]$$


Dataplane program
1 Segment
Segmented program
A
B C
D E
Abstract Program
2
3
Abstract Resource Semantics
Planner
4
Network Description
5 Planning Objectives
6 Plan
Annotated program
A
B F C
D E
Allocation model
Control program profile
7 Split
8 Segment mapping
A + F + D + E
9
A → F
F → D
F → E
D → E

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)

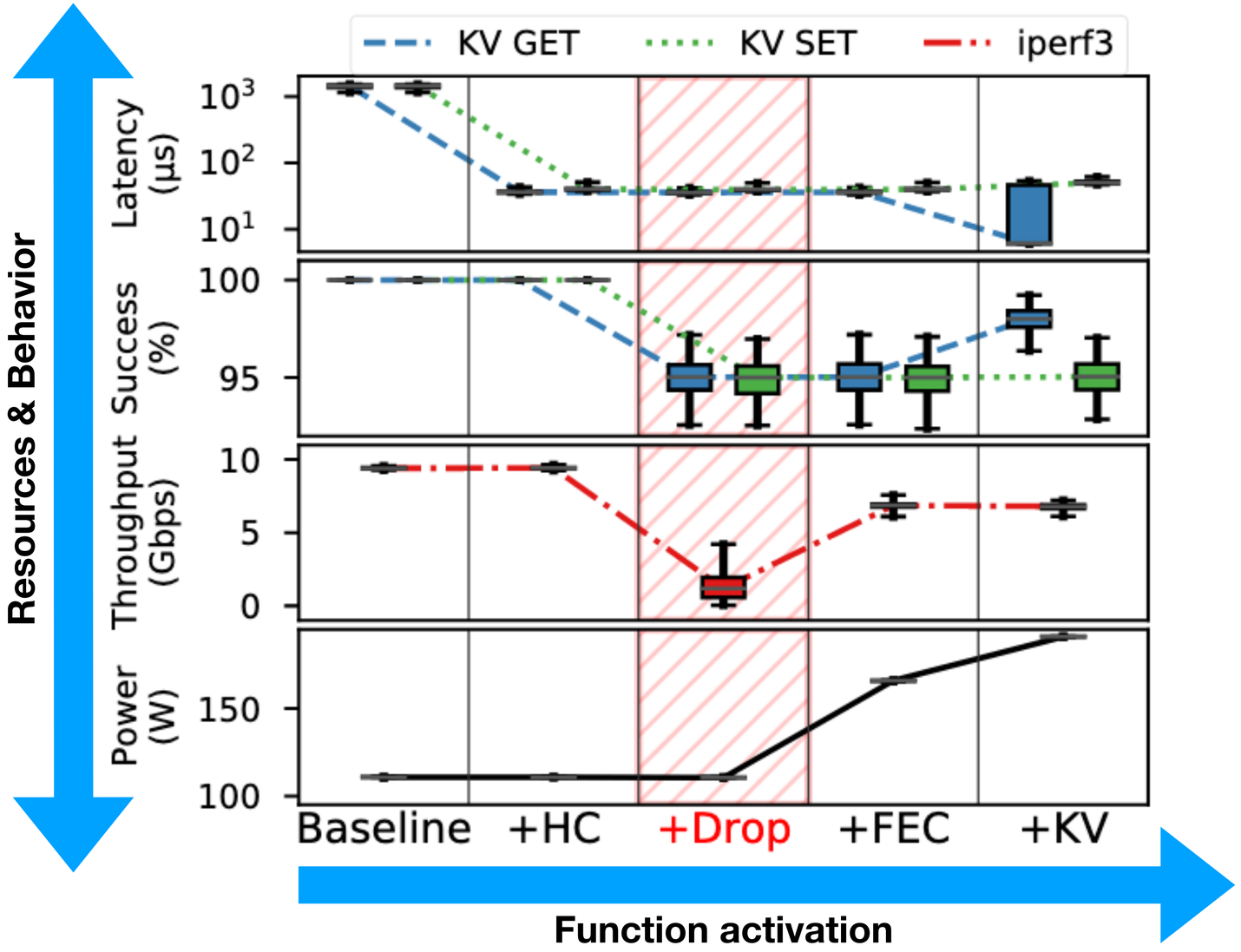
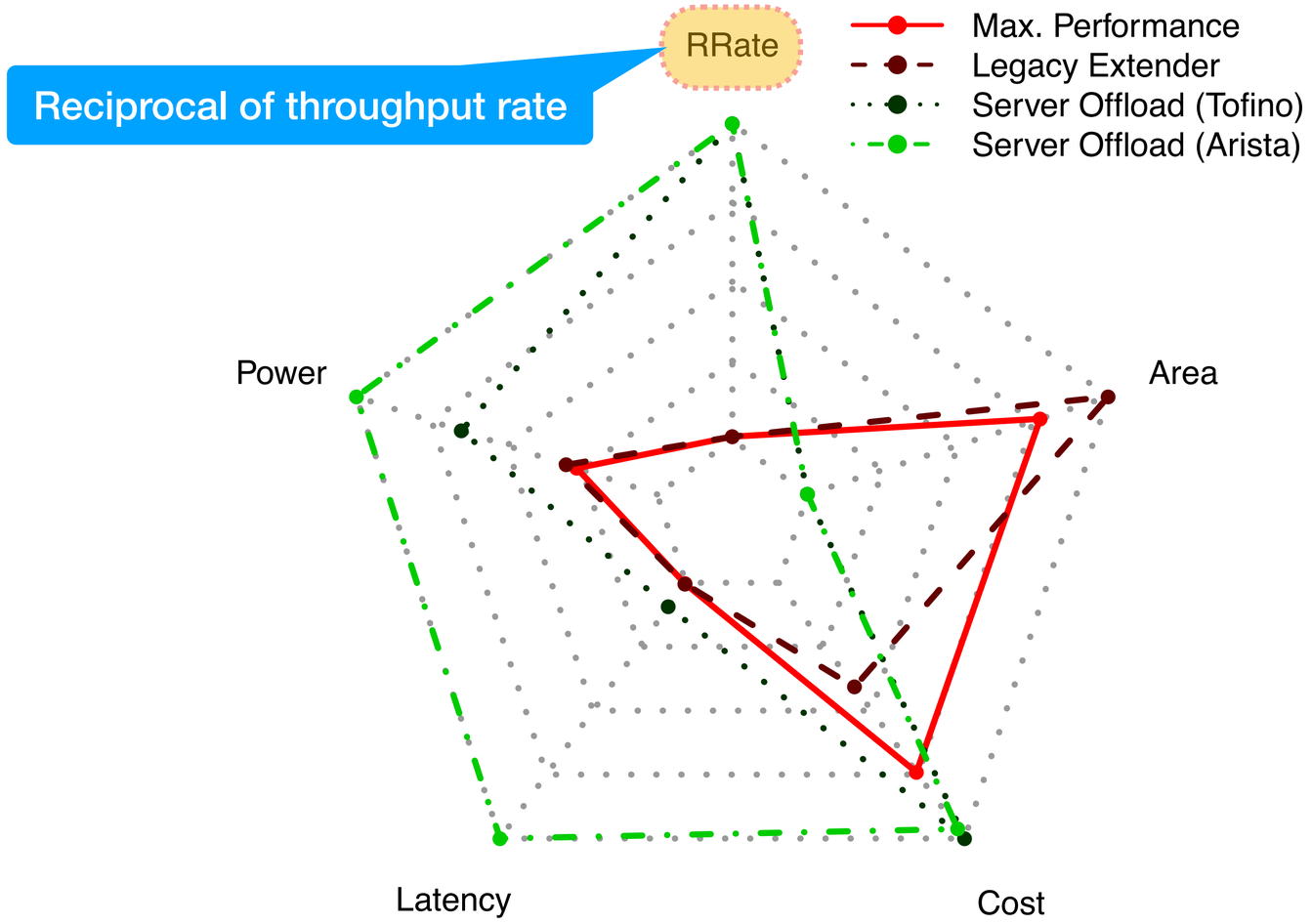


Fig 9: Plan comparison (Testbed + Simulation)



(Inner is better)

Fig 7: Multiple Programs vs Runtimes vs Splits in same network (Simulation)

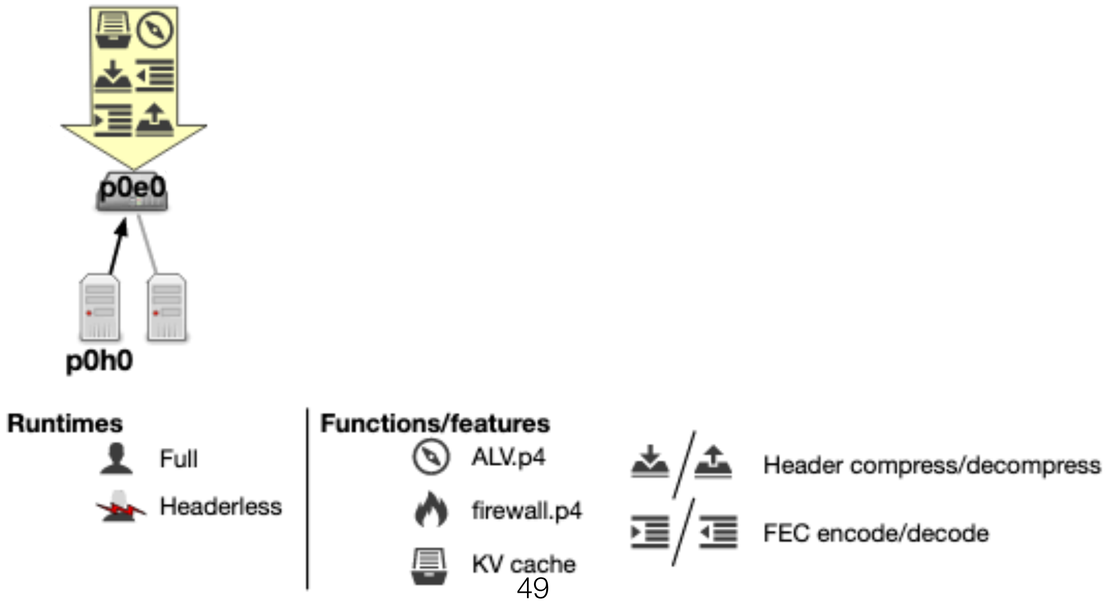


Fig 7: Multiple Programs vs Runtimes vs Splits in same network (Simulation)

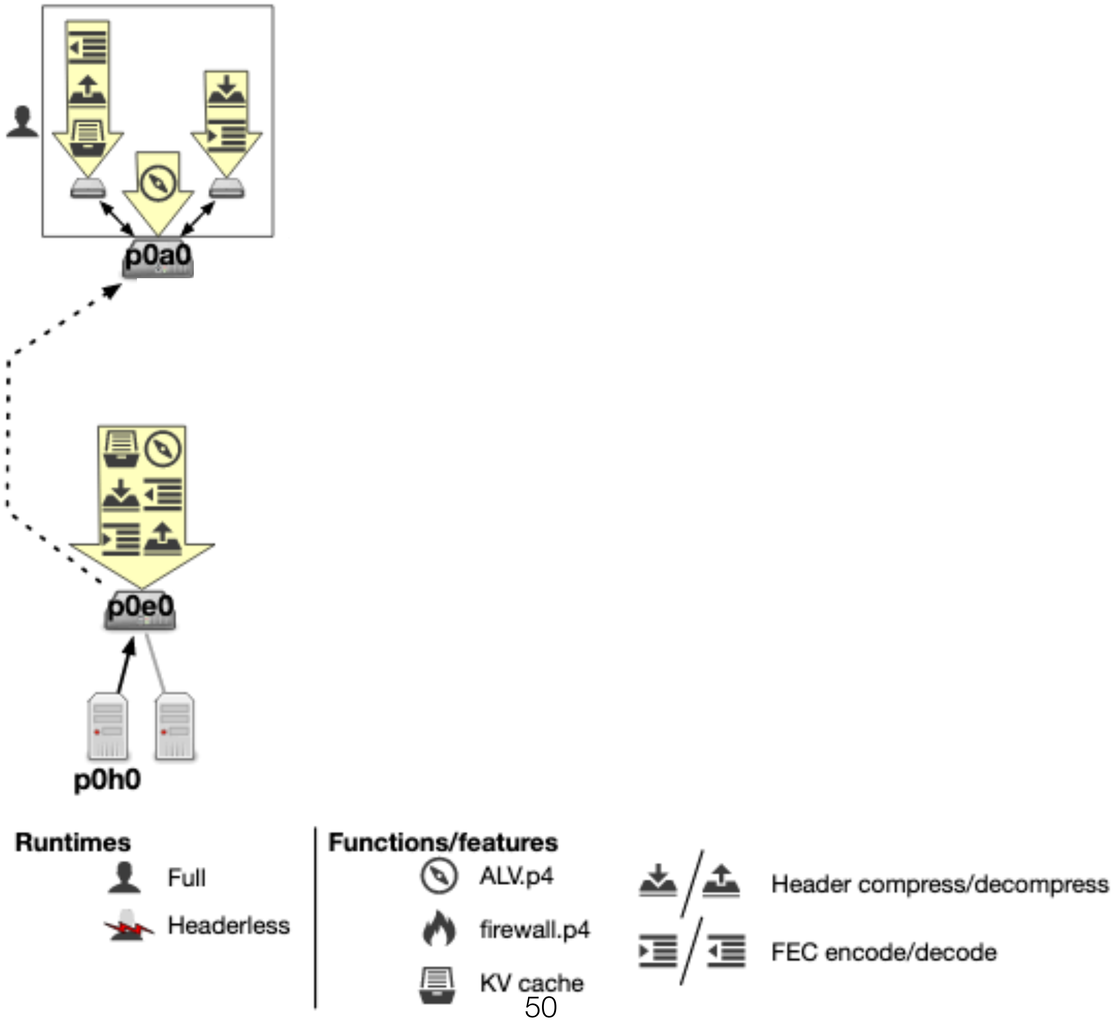
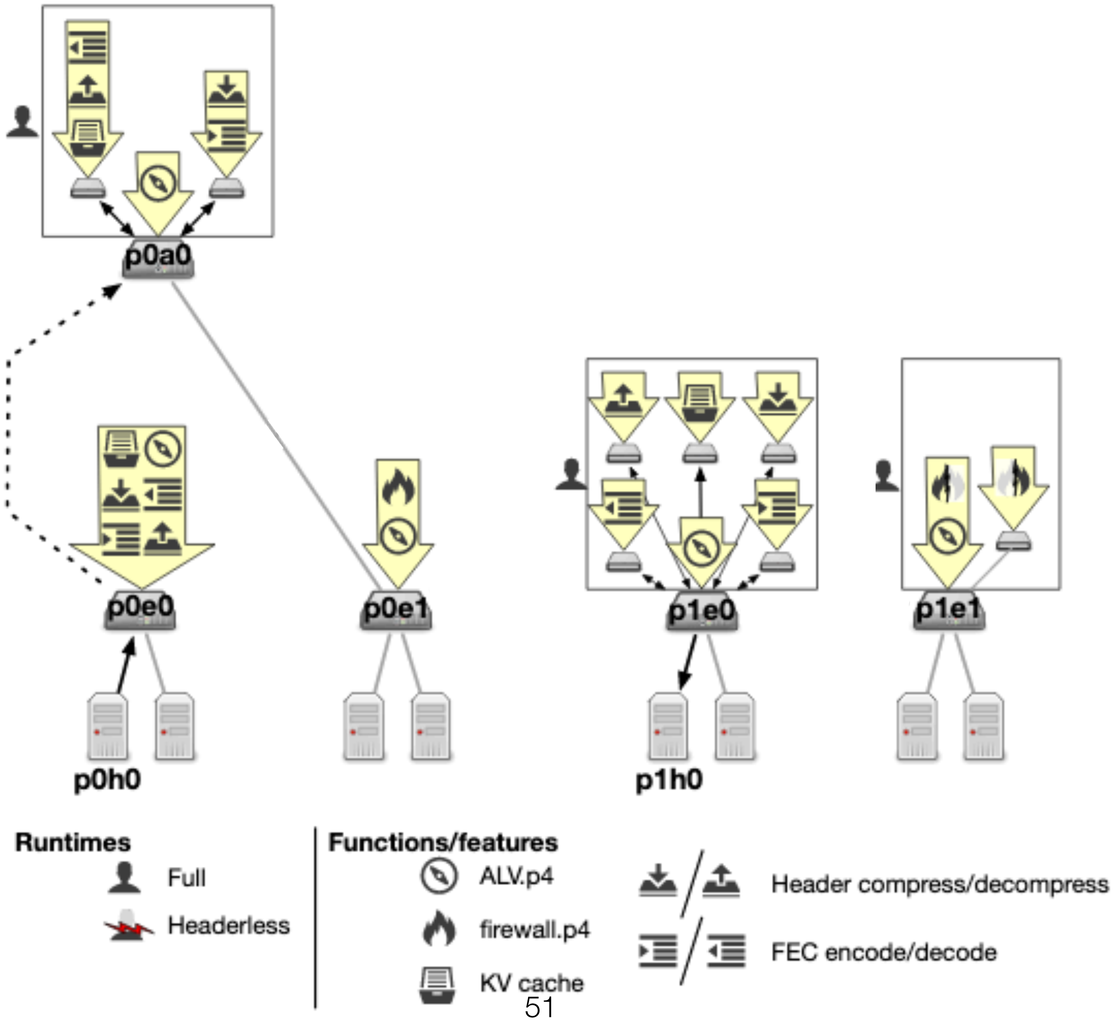
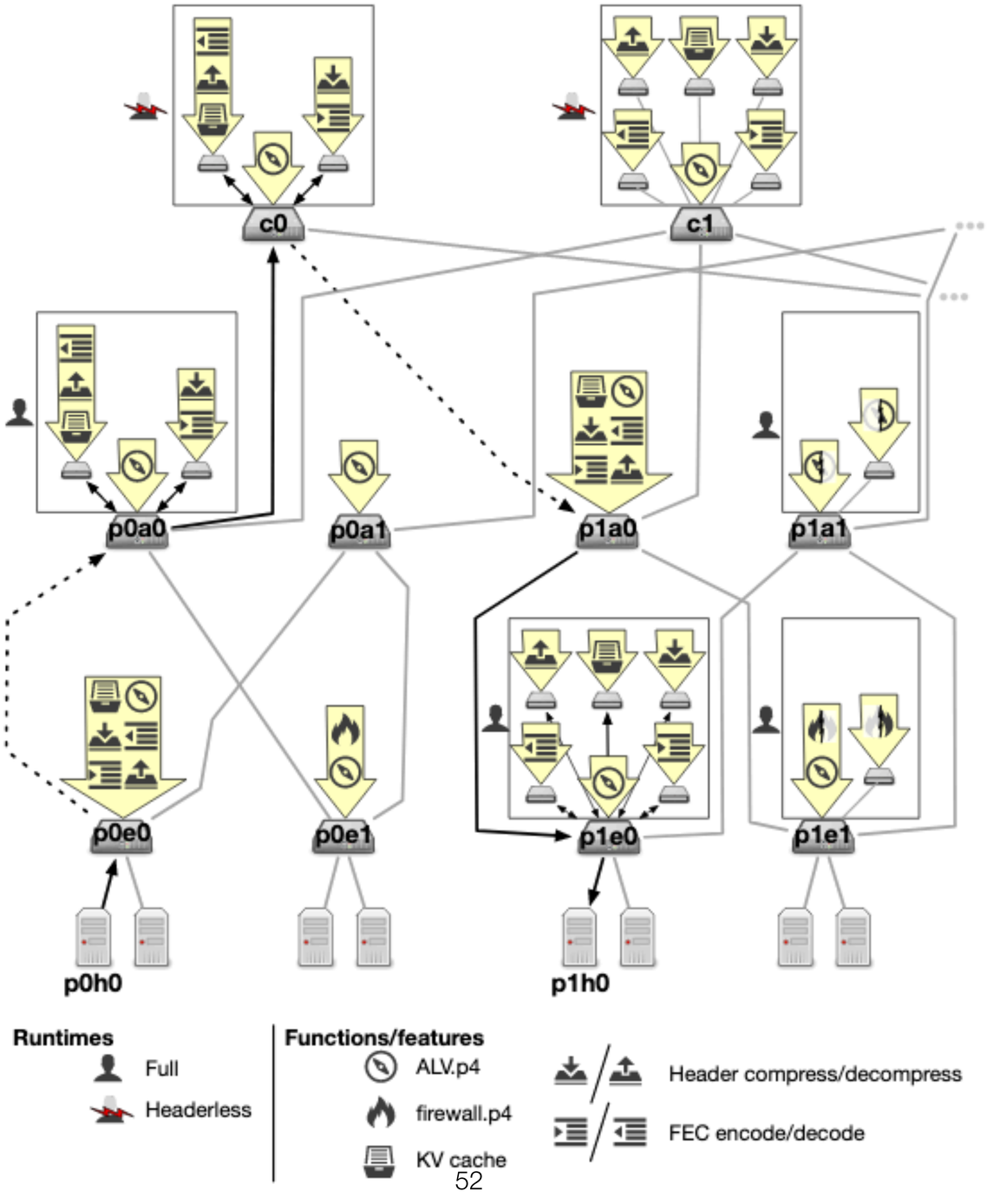


Fig 7: Multiple Programs vs Runtimes vs Splits in same network (Simulation)




c0
c1
p0a0
p0a1
p1a0
p1a1
p0e0
p0e1
p1e0
p1e1
p0h0
p1h0
Runtimes
Full
Headerless
Functions/features
ALV.p4
firewall.p4
KV cache
Header compress/decompress
FEC encode/decode

Flightplan demo

- **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