Flightplan: Dataplane Disaggregation and Placement for P4 programs

Nik Sultana

Illinois Tech

http://www.cs.iit.edu/~nsultana1

COIN Interim // 10th Feb 2022

Ack

Shivani Burad

Anirudh Chelluri

André DeHon

Hans Giesen

Zhaoyang Han

Latha A. Kant

Boon Thau Loo

Tony McAuley

Heena Nagda

Rakesh Nagda

Isaac Pedisich

Alexander Poylisher

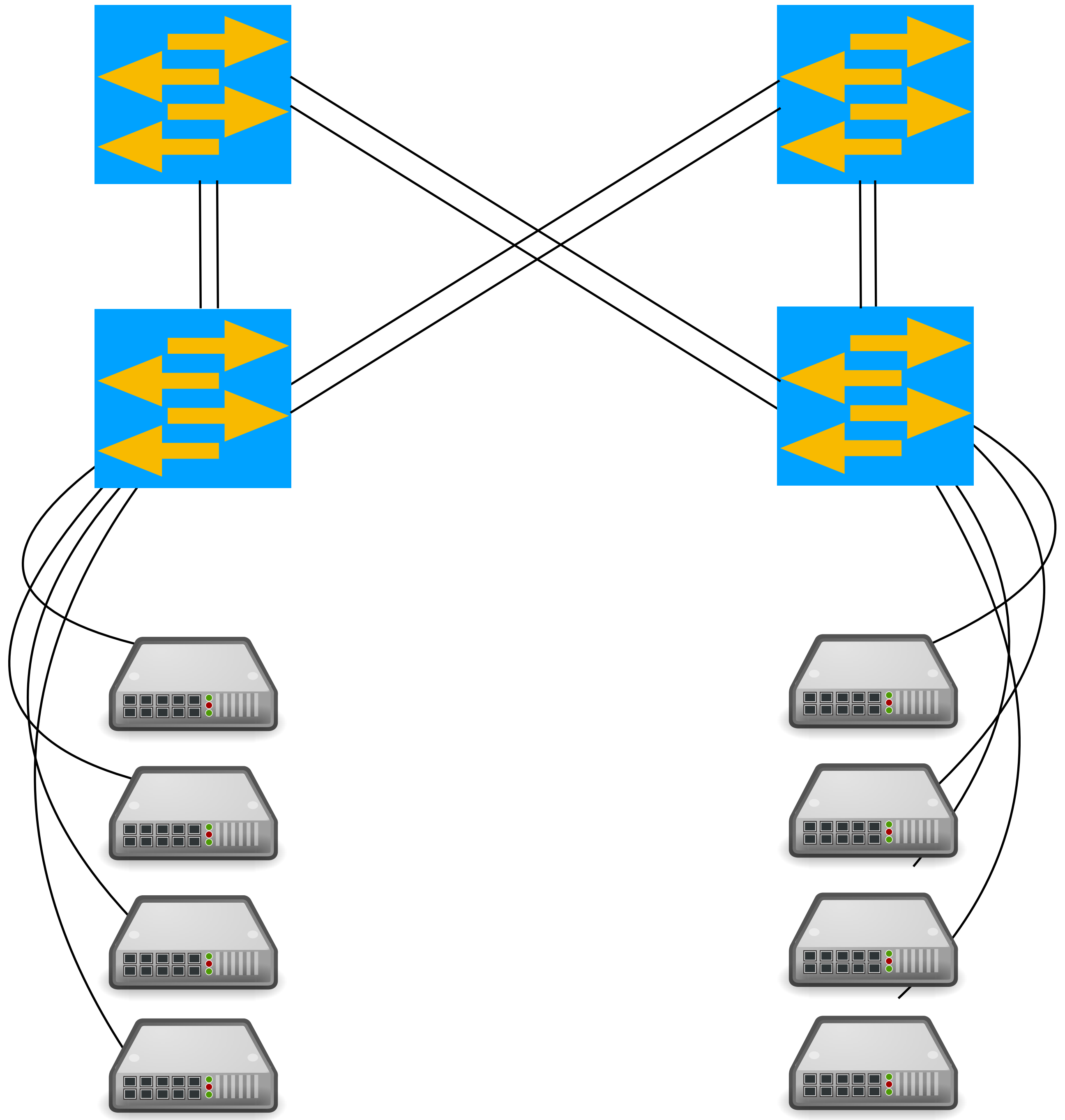
Nishanth Prabhu

Lei Shi

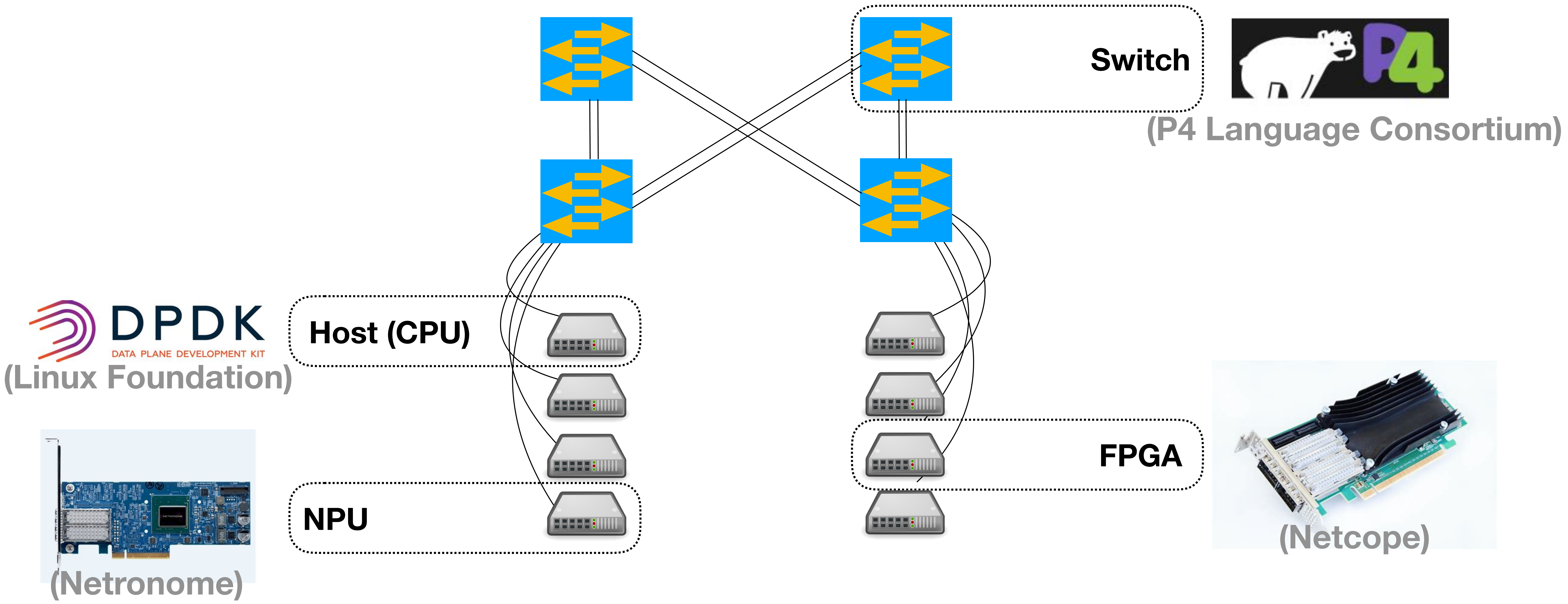
Nishanth Shyamkumar

John Sonchack

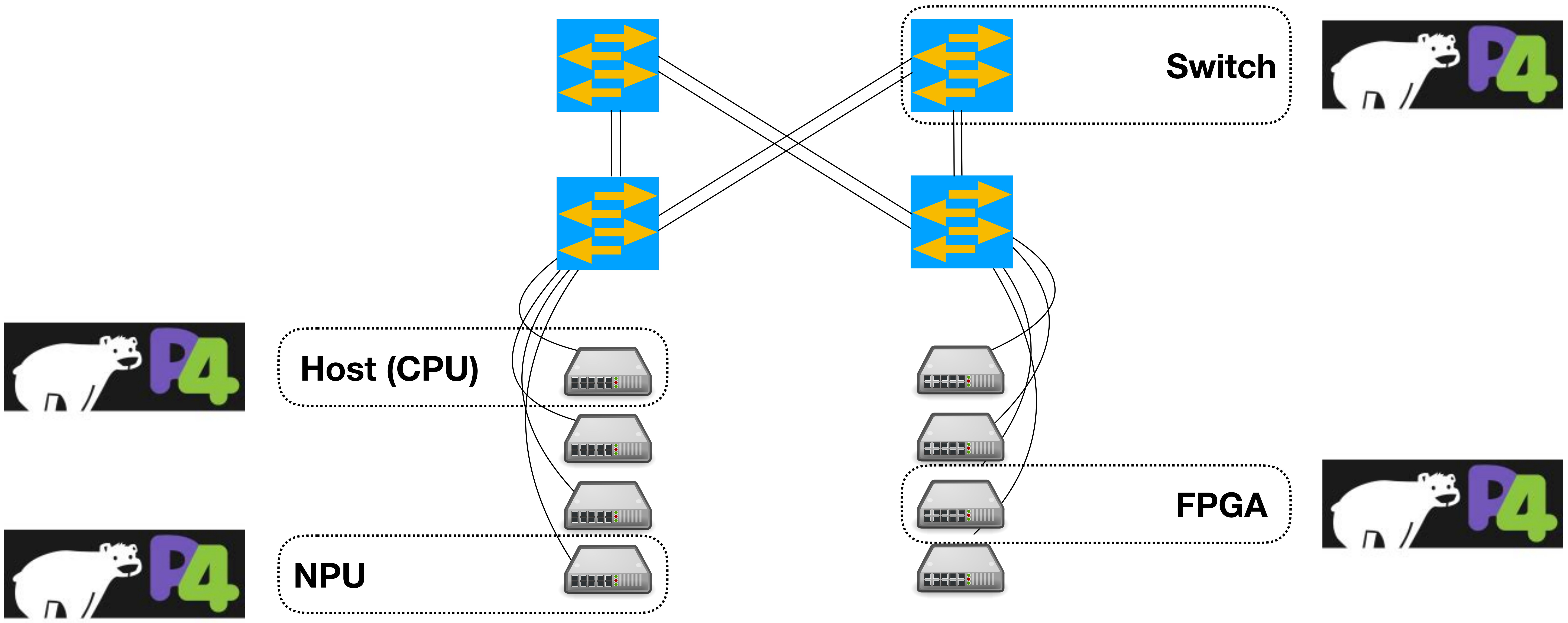
Dataplane Programmability
& In-Network Programming



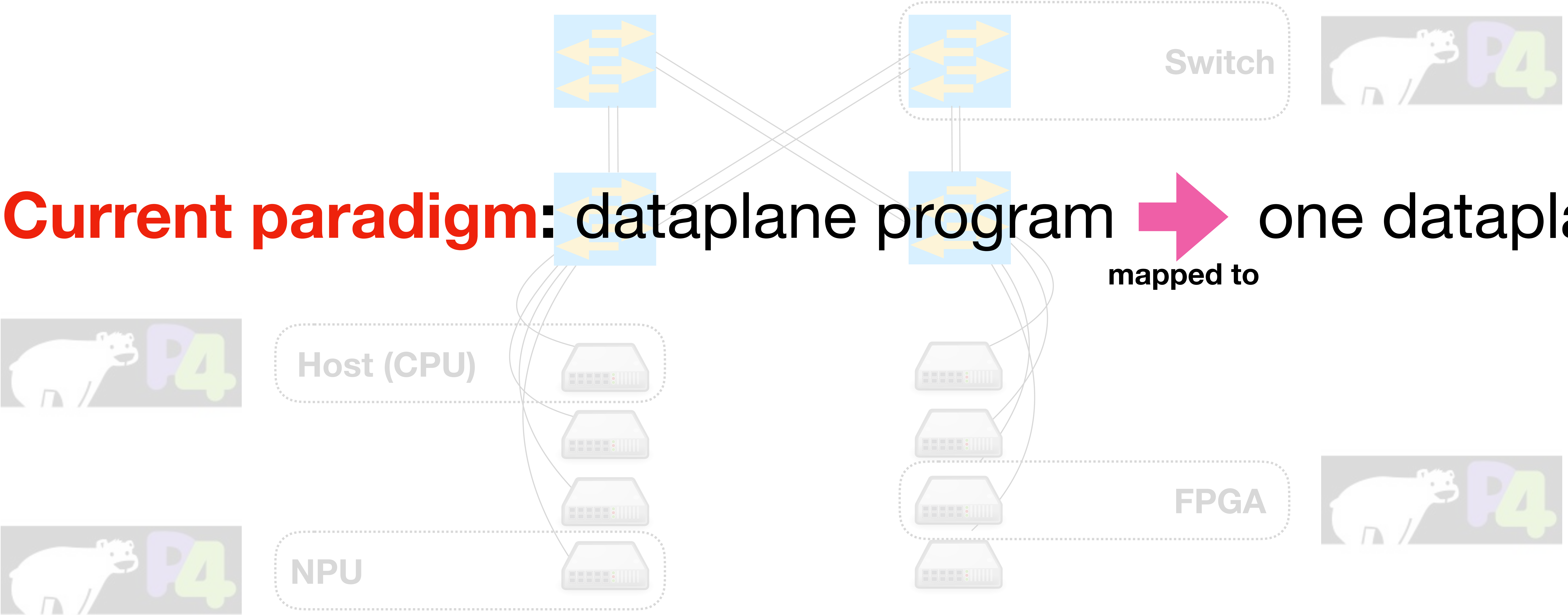
Dataplane Programmability
& In-Network Programming



Dataplane Programmability



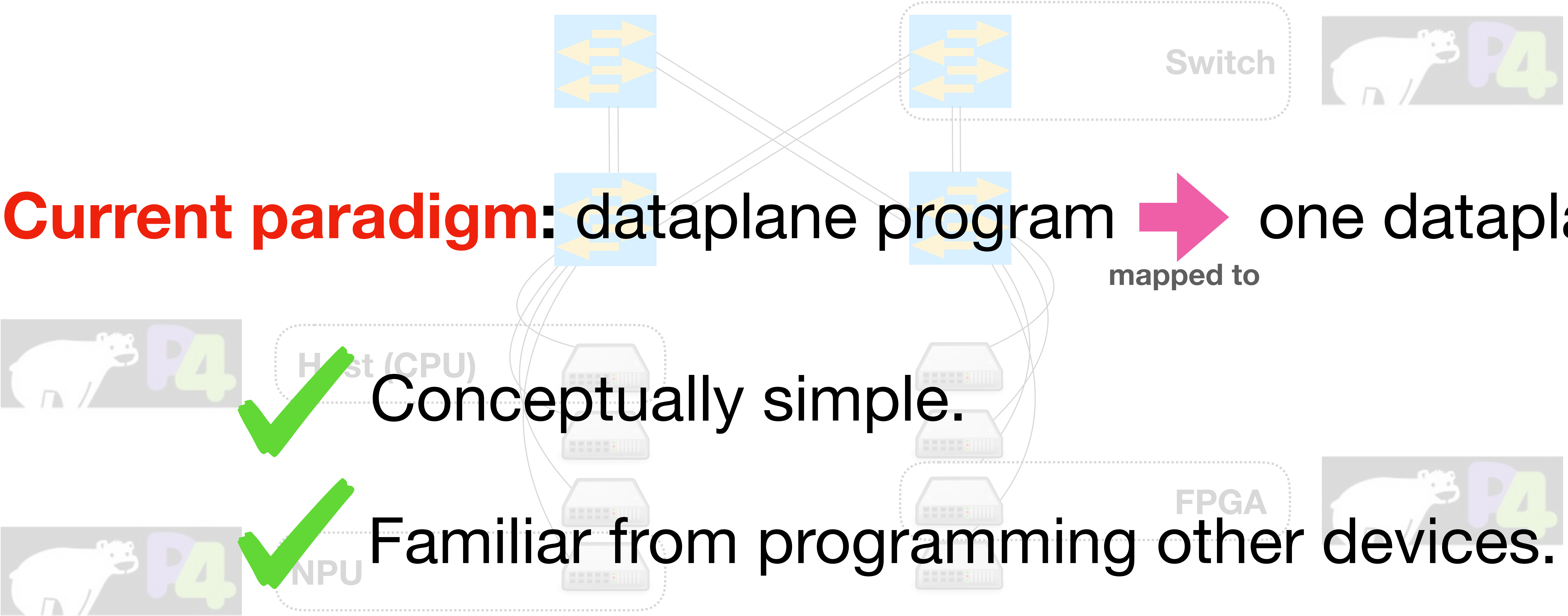
Dataplane Programmability



Current paradigm: dataplane program → one dataplane (device)

mapped to

Dataplane Programmability



Current paradigm: dataplane program ➡ one dataplane (device)

mapped to

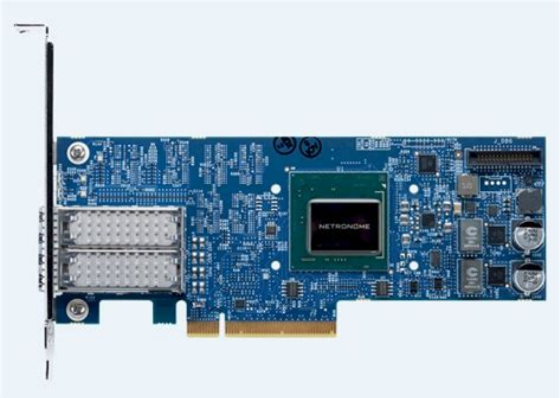
✓ Conceptually simple.

✓ Familiar from programming other devices.

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

Current paradigm: dataplane program → one dataplane

mapped to

xor DPDK DATA PLANE DEVELOPMENT KIT xor  xor 

Exclusive OR

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 DPDK xor  xor 

New paradigm:

dataplane program → suitable mix of dataplanes

(and ) or DPDK or (and DPDK) or

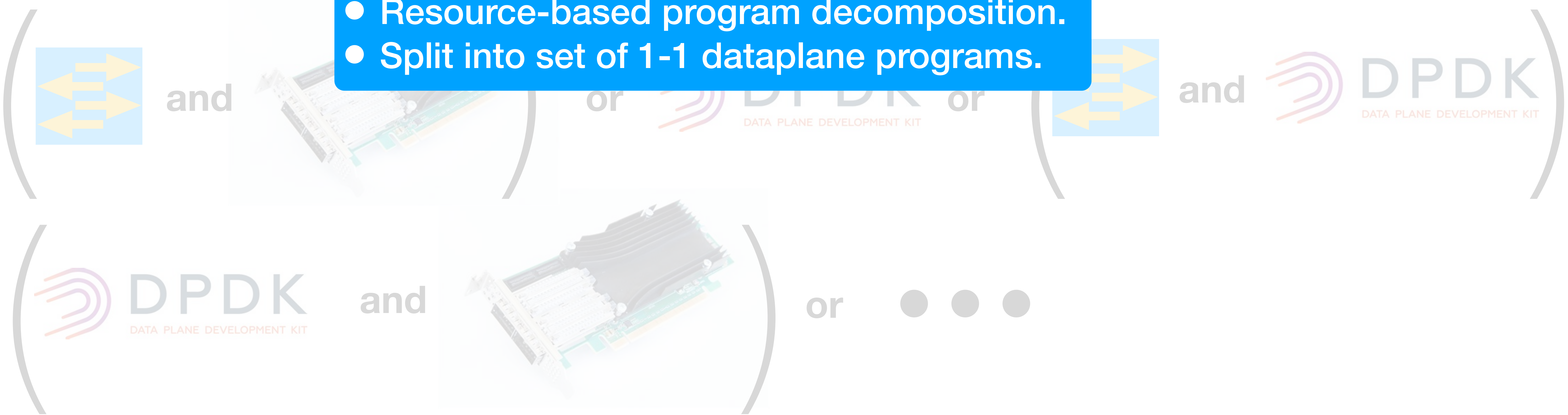
(DPDK and ) or • • •

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.



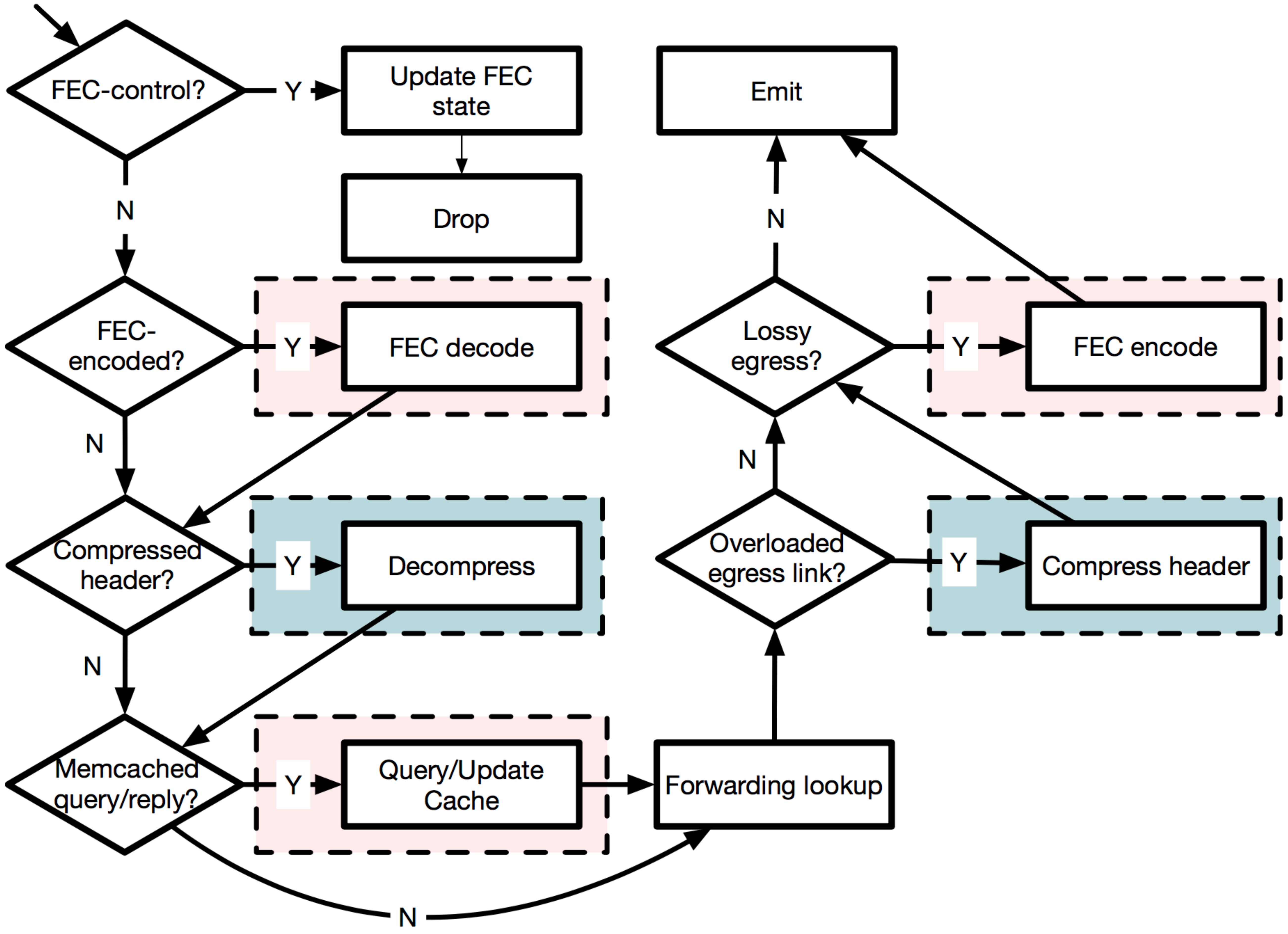
Current paradigm: dataplane program ➔ one dataplane

New paradigm:
dataplane program ➔ suitable mix of dataplanes

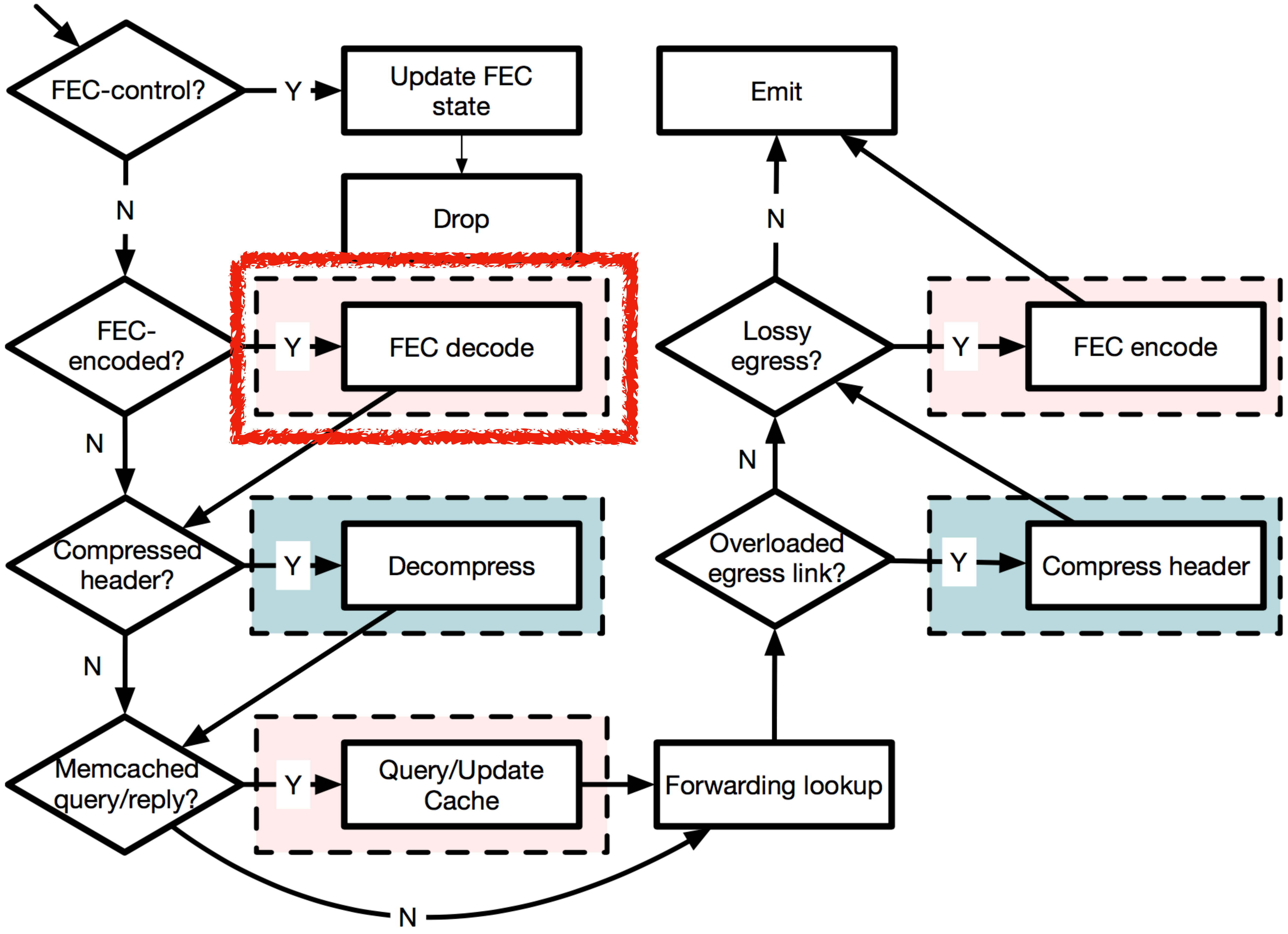
- Resource-based program decomposition.
- Split into set of 1-1 dataplane programs.

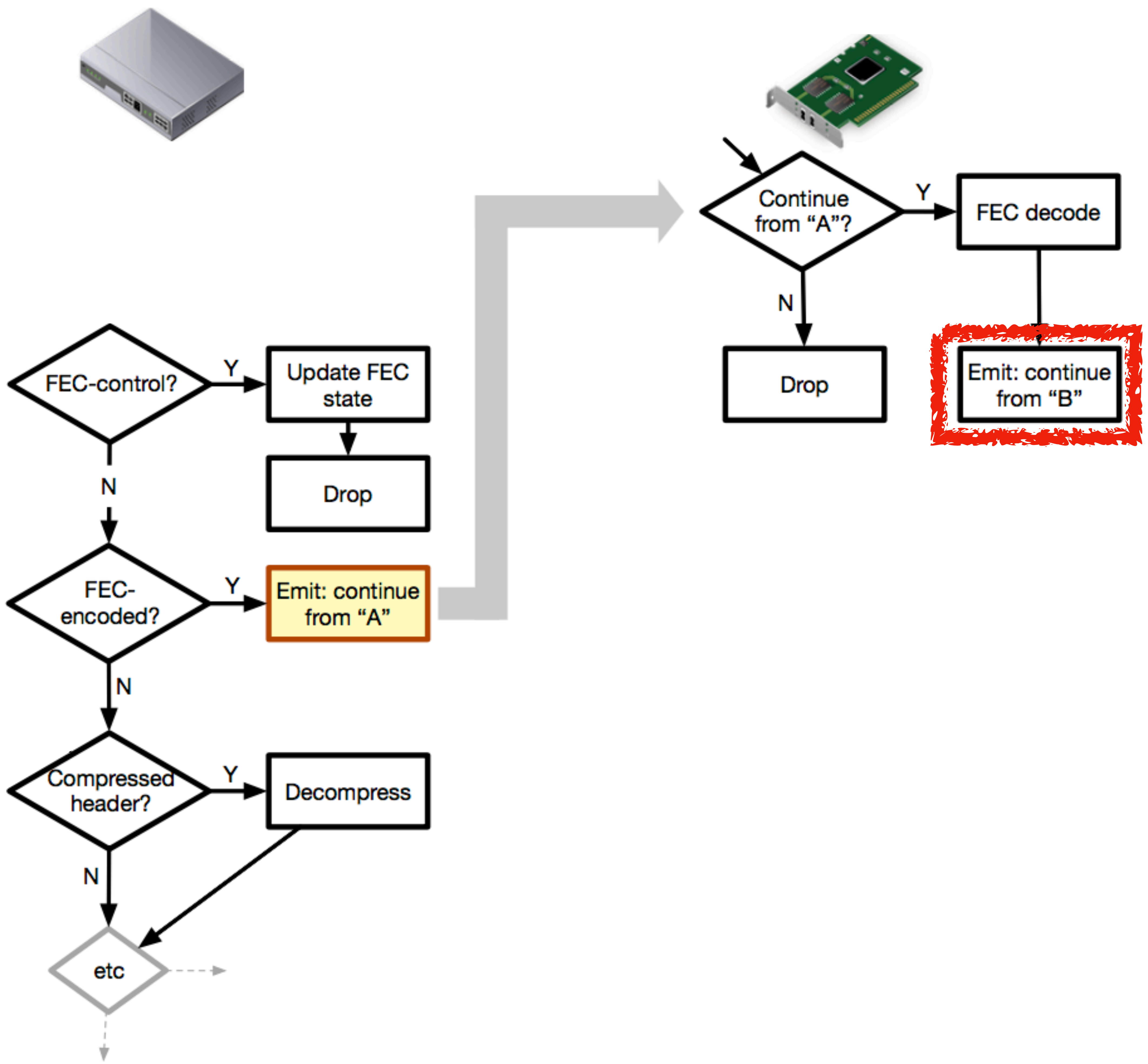
✔ Uses existing vendor toolchains, language & hardware.

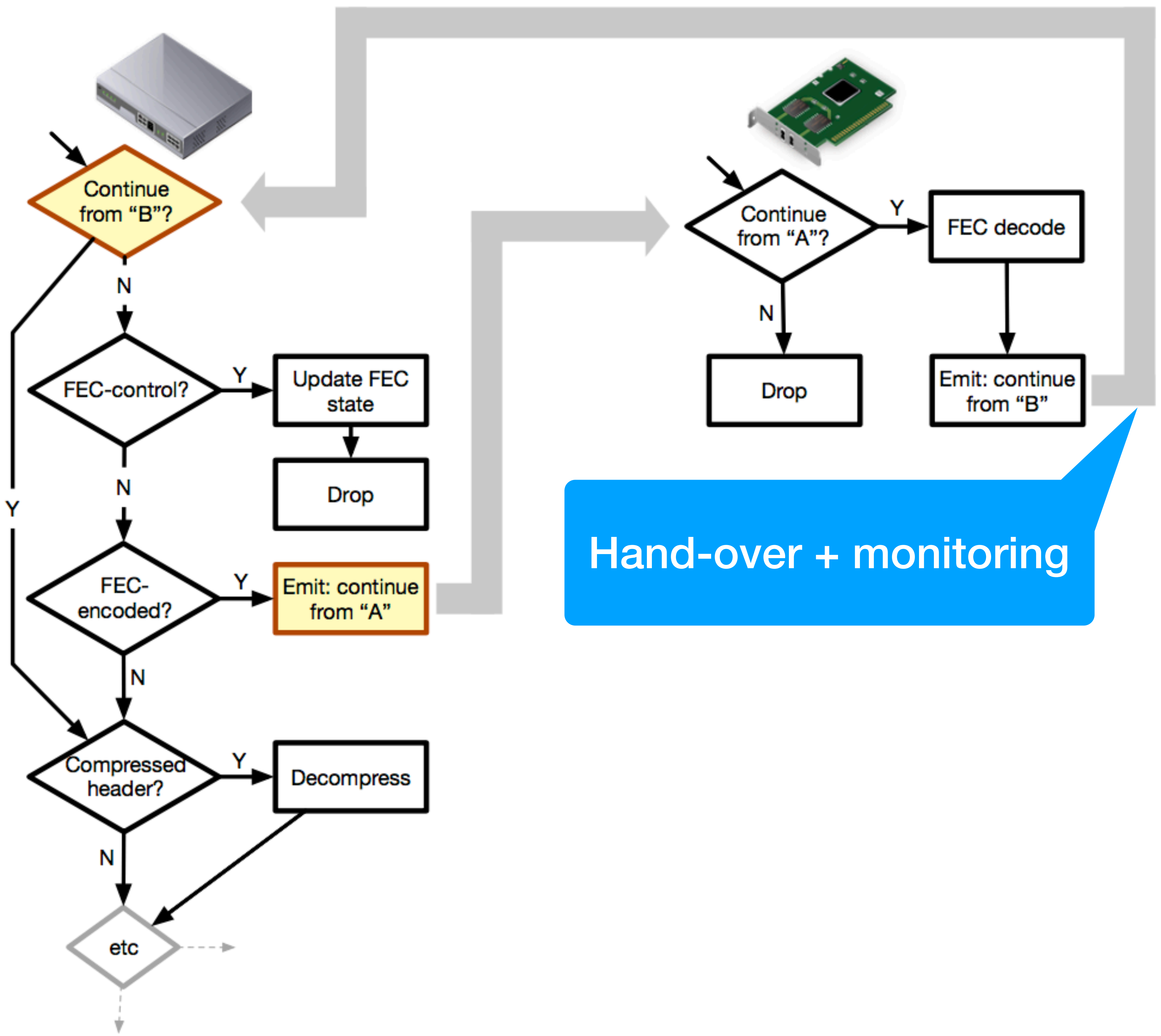
Example: “Crosspod” Program



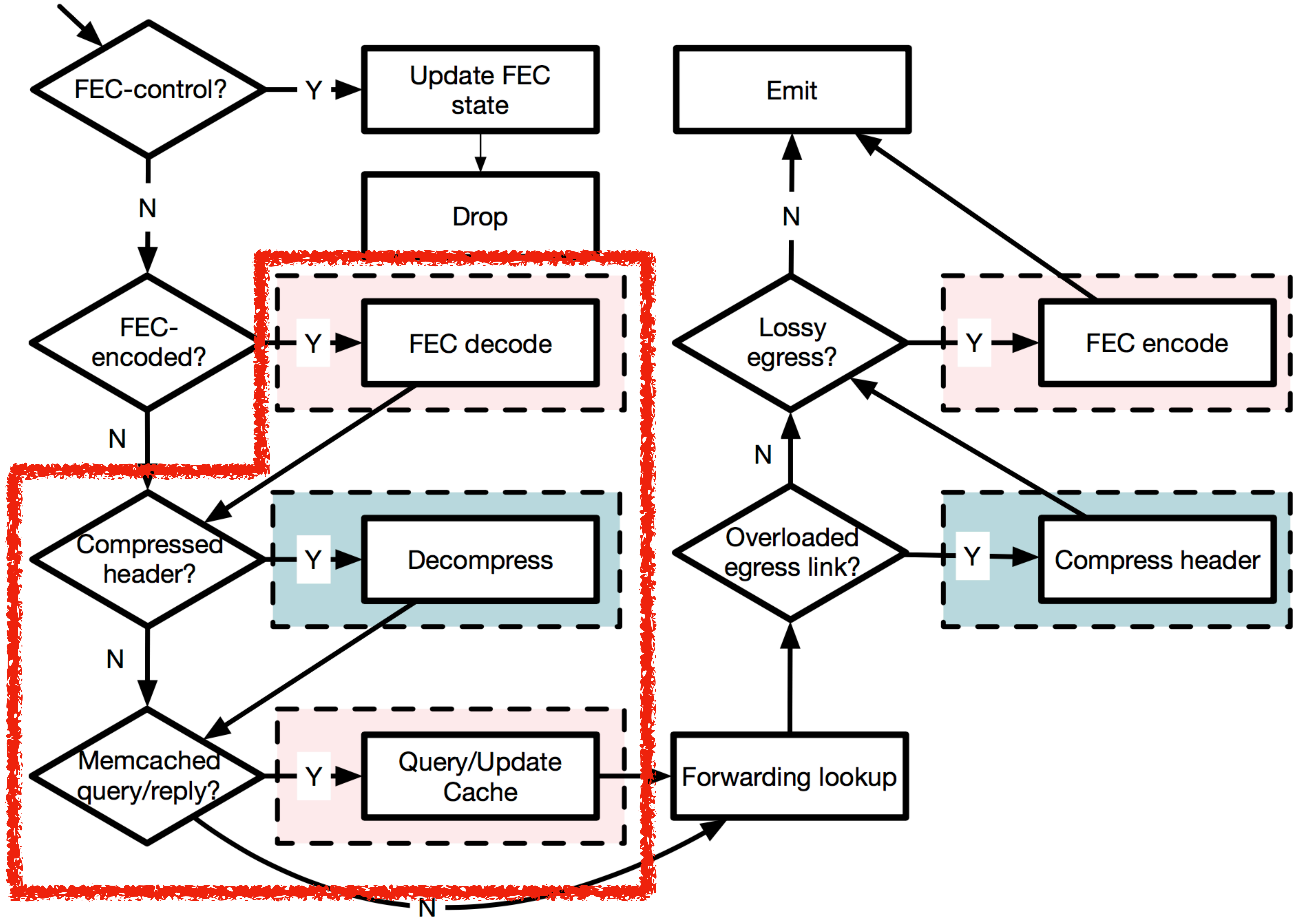
Example: “Crosspod” Program




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
Y
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"
Hand-over + monitoring


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

Current paradigm: dataplane program → one dataplane

New paradigm: **Dataplane Disaggregation**

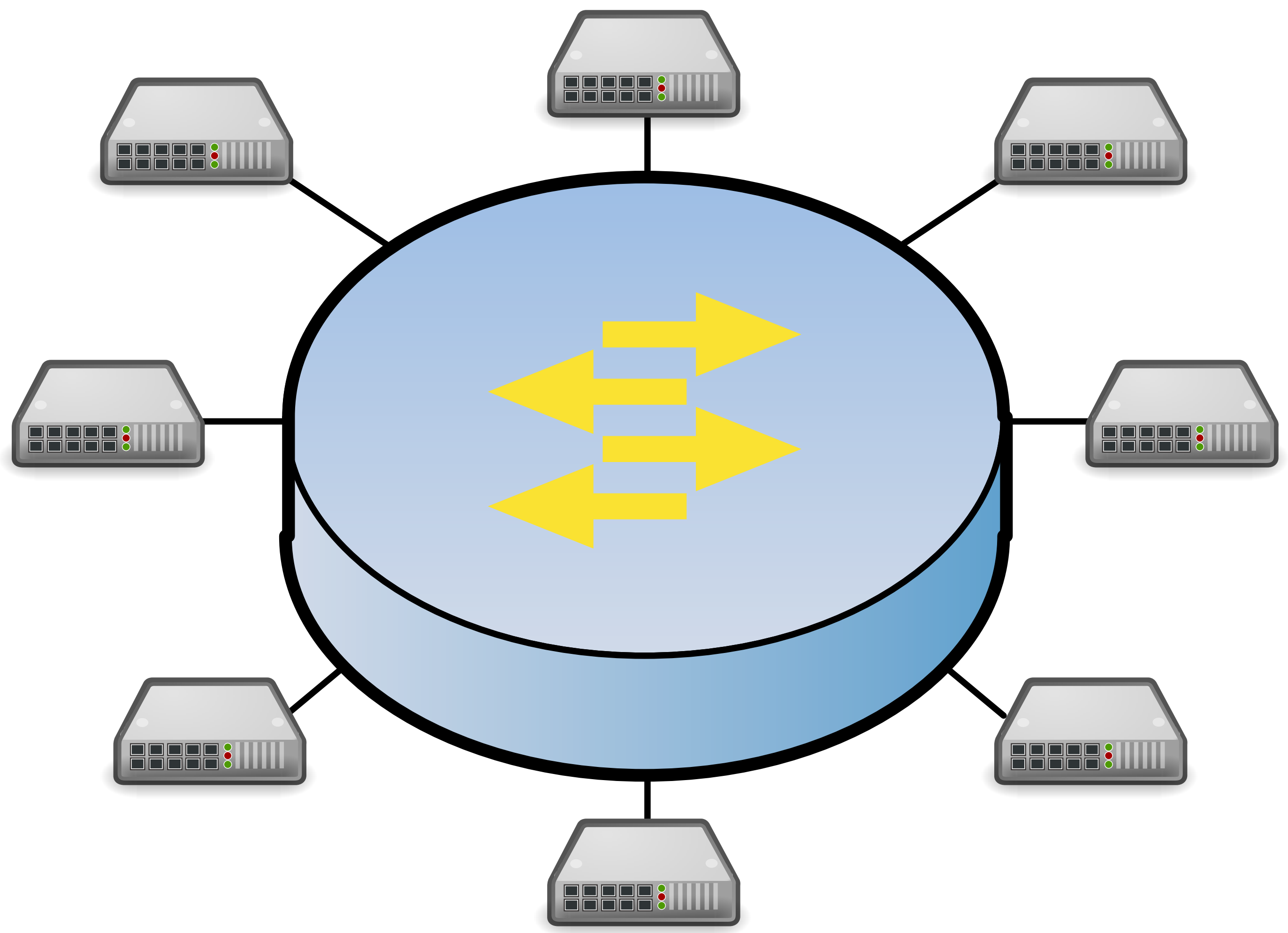
dataplane program → suitable mix of dataplanes

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

Dataplane Disaggregation

(≠ Server Disaggregation)
(≠ Switch Disaggregation)

Virtual Dataplane ➔ Set of Physical Dataplanes



"One big switch" ➔ "One big *programmable* switch"

Current paradigm: dataplane program → one dataplane

xor xor

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.

and or or and or

✓ Uses existing vendor toolchains, language & hardware.

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

Current paradigm: dataplane program → one dataplane

xor xor xor

Dataplane Disaggregation:

dataplane program → suitable mix of dataplanes

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

(and) or DPDK or (and DPDK) or

(DPDK and) or • • •

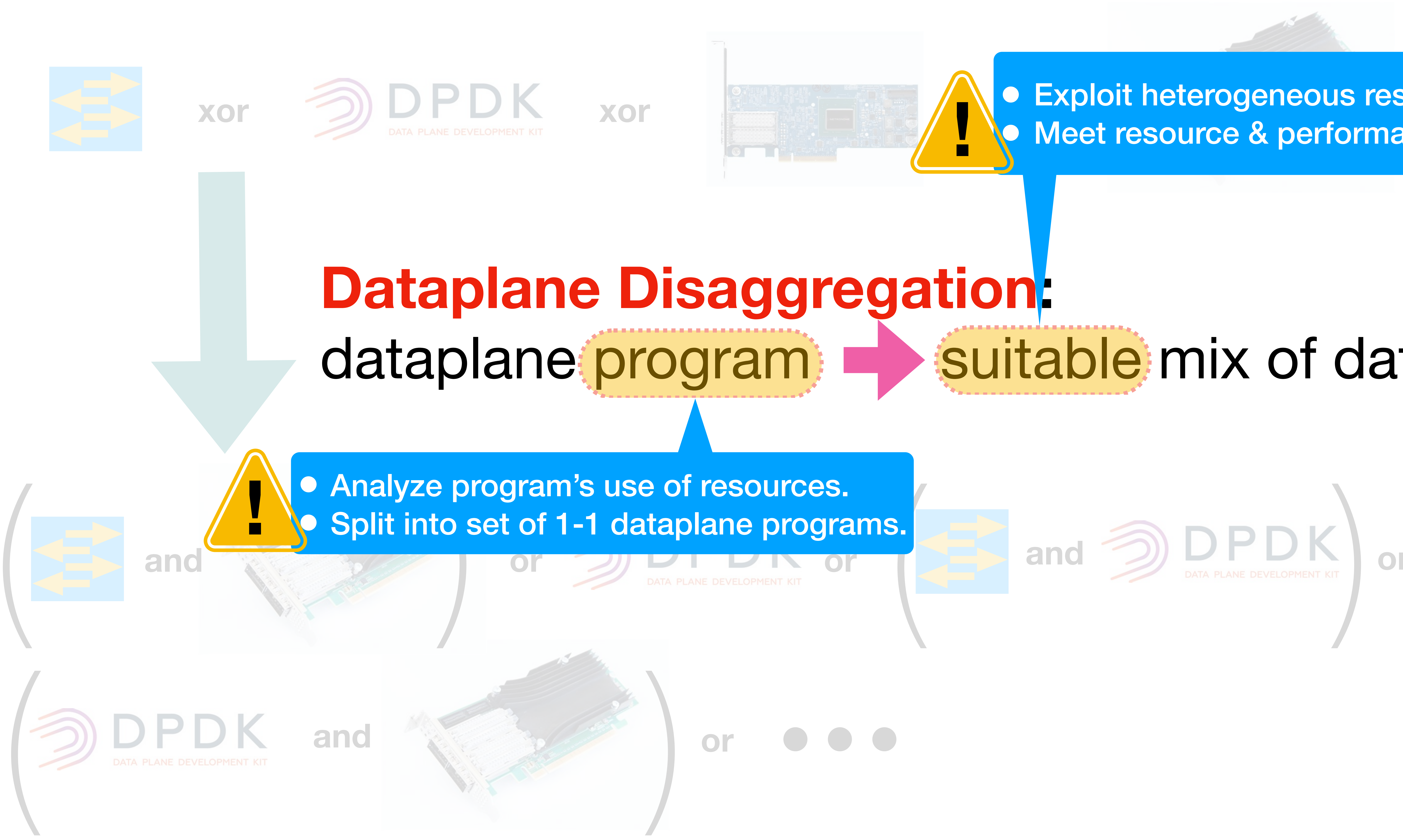
Current paradigm: dataplane program → one dataplane

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.



Current paradigm: dataplane program → one dataplane

xor DPDK xor

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

Placement

Dataplane Disaggregation:

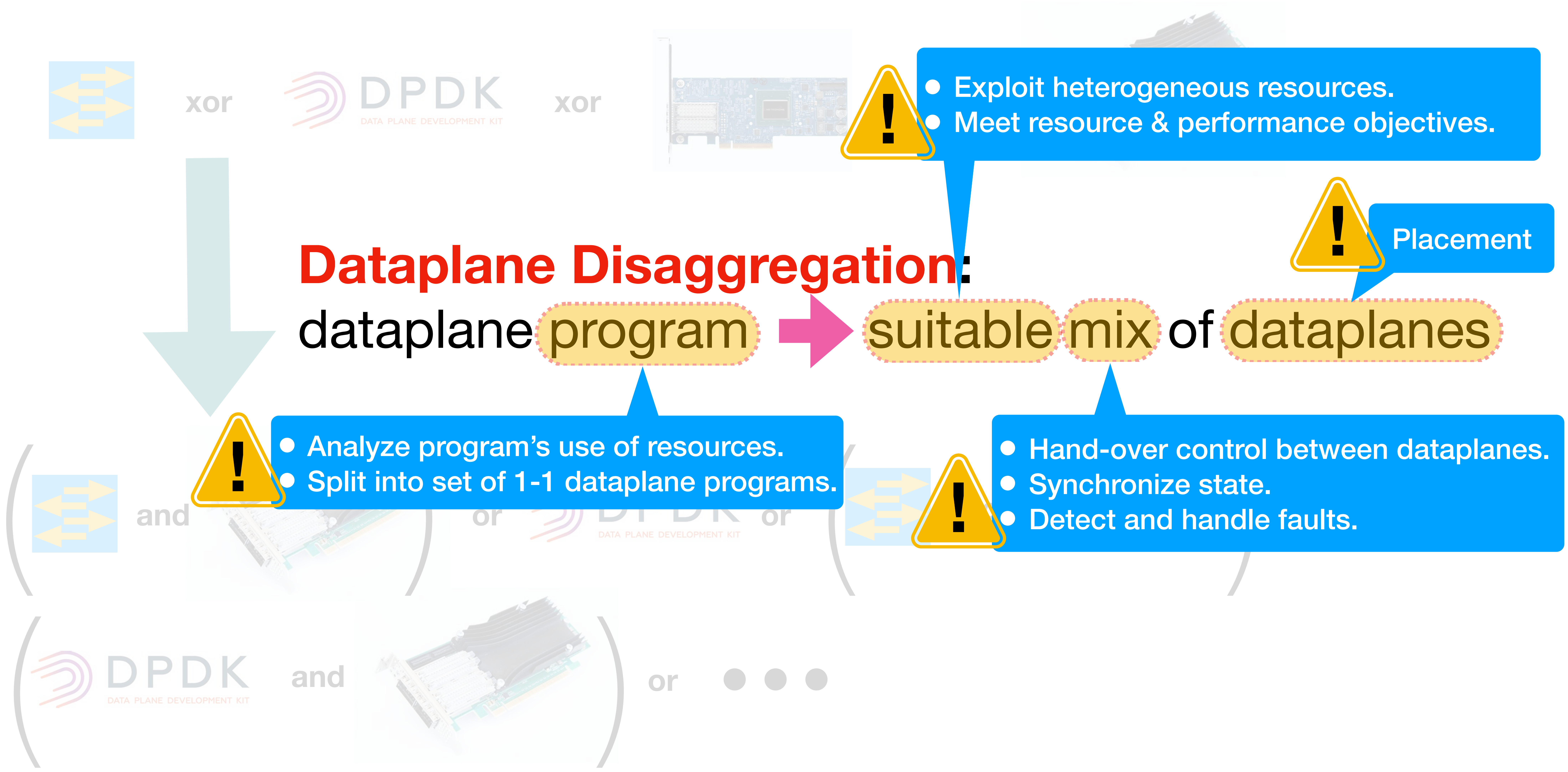
dataplane program → suitable mix of dataplanes

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

(and) or DPDK or (and DPDK) or

(DPDK and) or • • •

Current paradigm: dataplane program ➔ one dataplane



Current paradigm: dataplane program ➡ one dataplane

xor DPDK xor

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.

and or DPDK or

DPDK 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/

ogram one dataplane

xor DPDK xor

Flightplan

Dataplane Disaggregation:

dataplane program → suitable mix of dataplanes

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

Placement

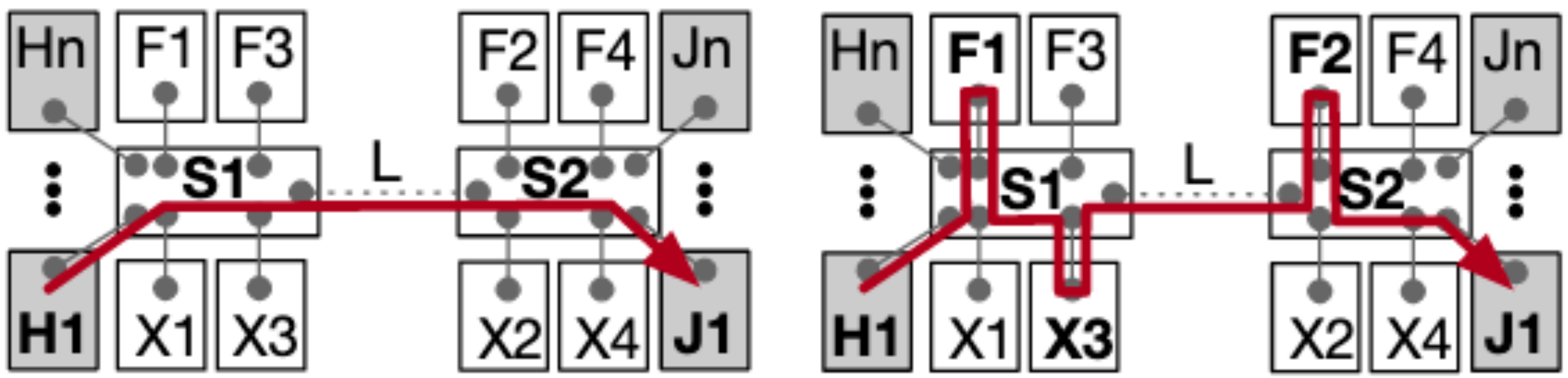
- Analyze program's use of resources.
- Split into set of 1-1 dataplane programs.

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

and or DPDK or

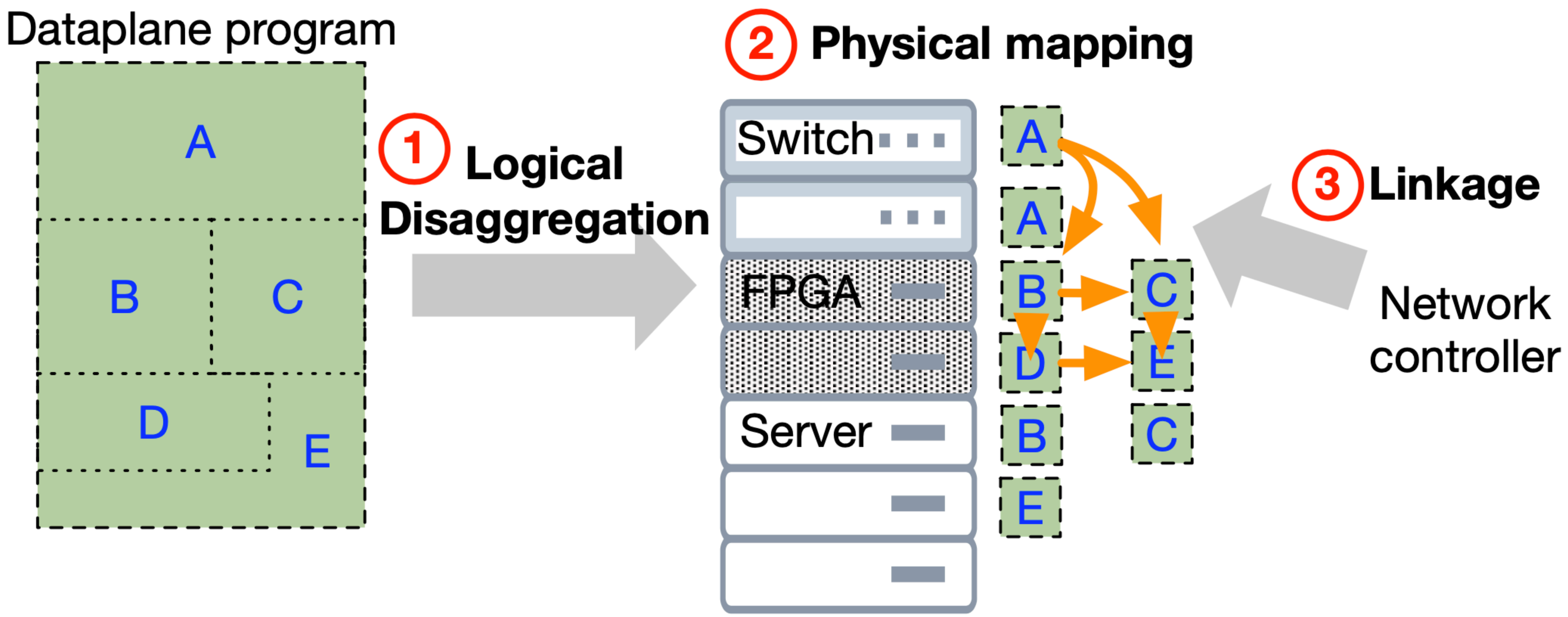
DPDK and

- 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.

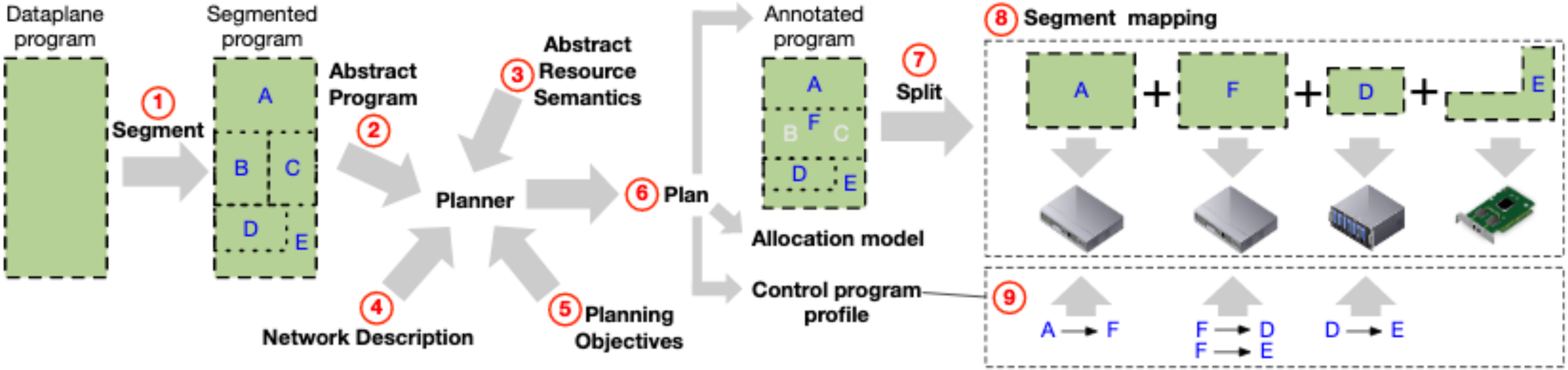


Hn
F1
F3
F2
F4
Jn
S1
L
S2
H1
X1
X3
X2
X4
J1
Hn
F1
F3
F2
F4
Jn
S1
L
S2
H1
X1
X3
X2
X4
J1

Flightplan


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

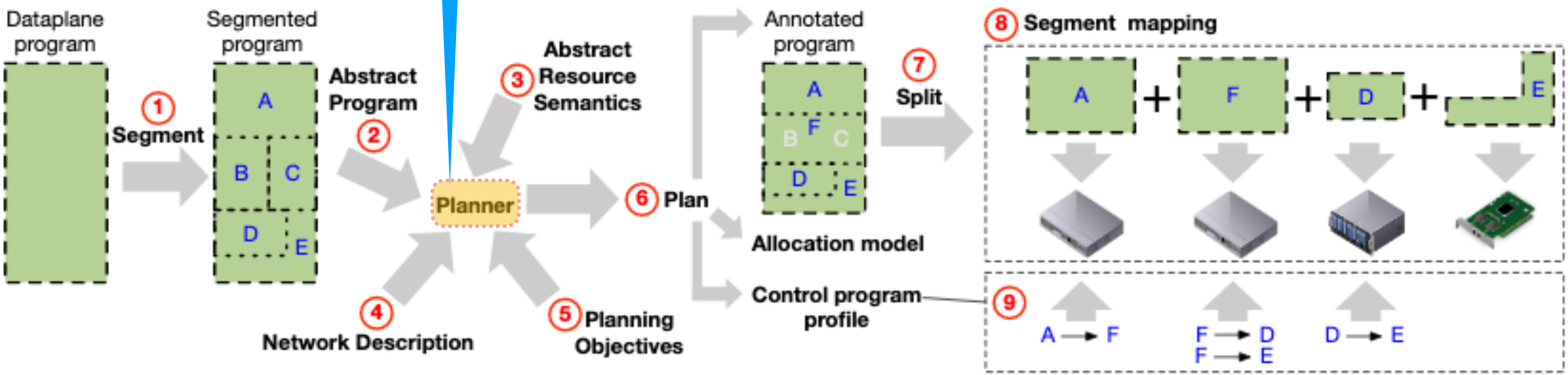
Flightplan



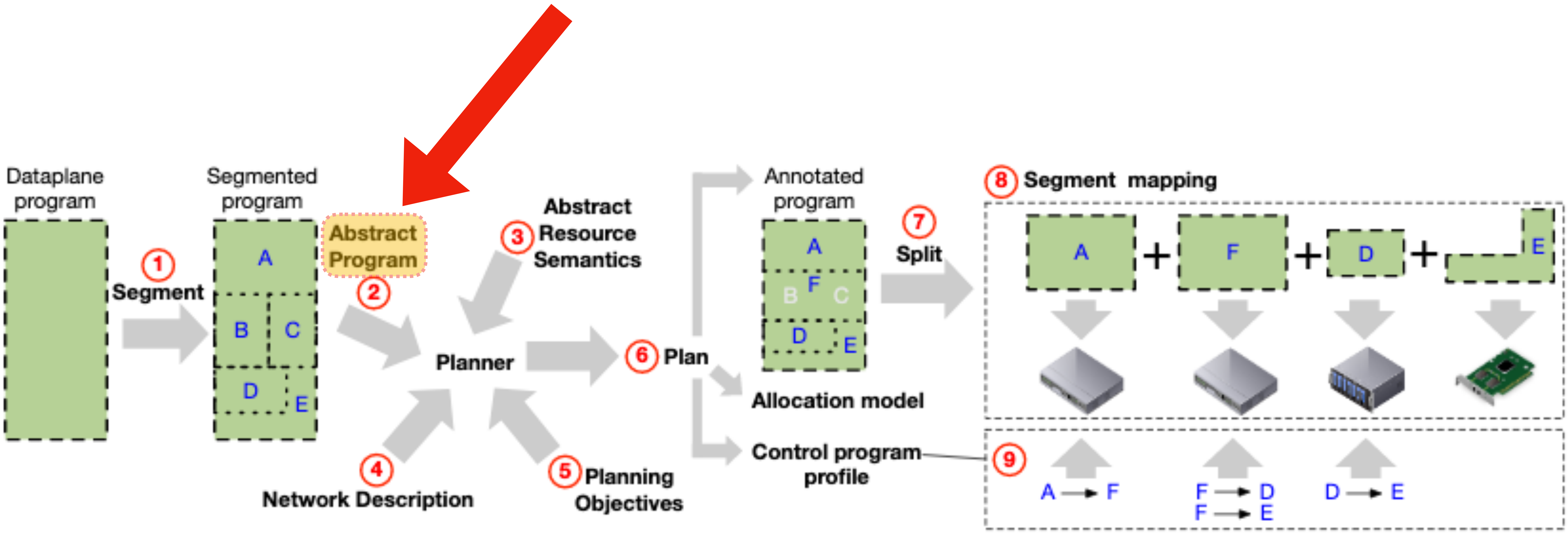
Dataplane program
Segmented program
A
B
C
D
E
1 Segment
Abstract Program
2
3 Abstract Resource Semantics
Planner
4 Network Description
5 Planning Objectives
6 Plan
Annotated program
A
B F C
D
E
7 Split
Allocation model
Control program profile
8 Segment mapping
A + F + D + E
9
A → F
F → D
F → E
D → E

Flightplan

Idea from symbolic AI: rule-based search.



Flightplan



Dataplane program
Segment
Segmented program
A
B
C
D
E
Abstract Program
Abstract Resource Semantics
Planner
Network Description
Planning Objectives
Plan
Annotated program
A
F
B
C
D
E
Allocation model
Control program profile
Split
Segment mapping
A
F
D
E
A → F
F → D
F → E
D → E

Flightplan

Example program (Crosspod)

Segment annotation

Resource dependence

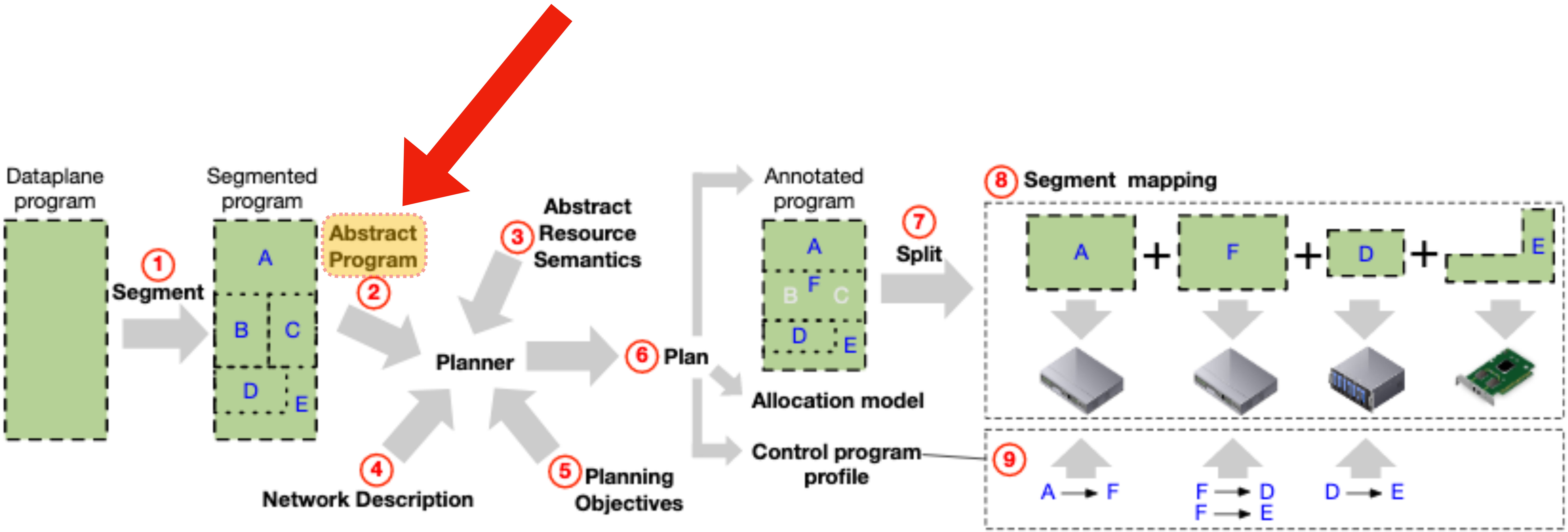
```
bit<1> compressed_link = 0;
bit<1> run_fec_egress = 0;
...
flyto(Compress);
// If heading out on a multiplexed link, then header compress.
egress_compression.apply(meta.egress_spec, compressed_link);
if (compressed_link == 1) {
    header_compress(forward);
    if (forward == 0) {
        drop();
        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_ethertype = hdr.eth.type;
        FEC_ENCODE(hdr.fec, k, h);
        ...
```

Abstract program

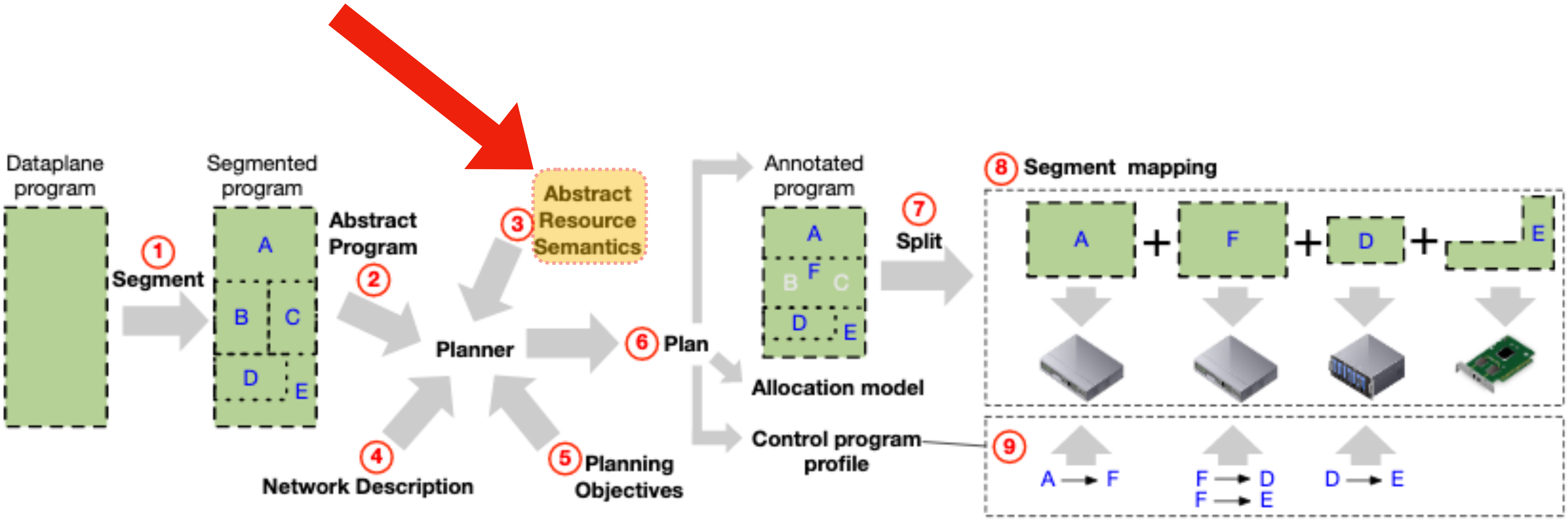
```
flyto(Compress);
// If heading out on a multiplexed link, then header compress.
egress_compression.apply(meta.egress_spec, compressed_link);
if (compressed_link == 1) {
    header_compress(forward);
    if (forward == 0) {
        drop();
        return;
    }
}
flyto(FEC_Encode);
```

(Rule generation is fully automatic)

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


Dataplane program
Segmented program
Abstract Program
Abstract Resource Semantics
Annotated program
Segment mapping
Segment
Planner
Plan
Split
Allocation model
Control program profile
Network Description
Planning Objectives
A
B
C
D
E
F
A → F
F → D
F → E
D → E

Flightplan


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

Flightplan

Abstract Resource Semantics

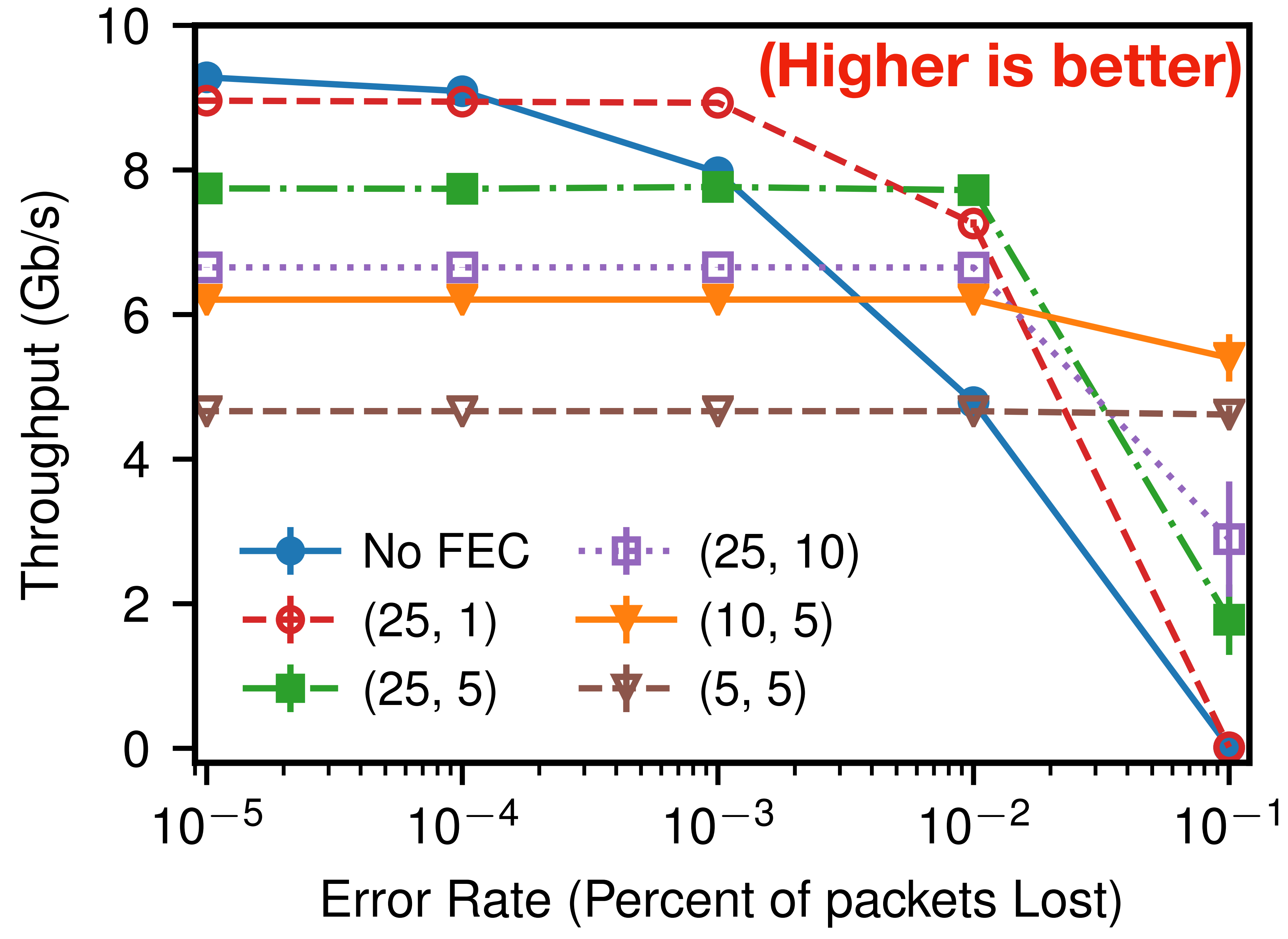
(Specific device + platform)

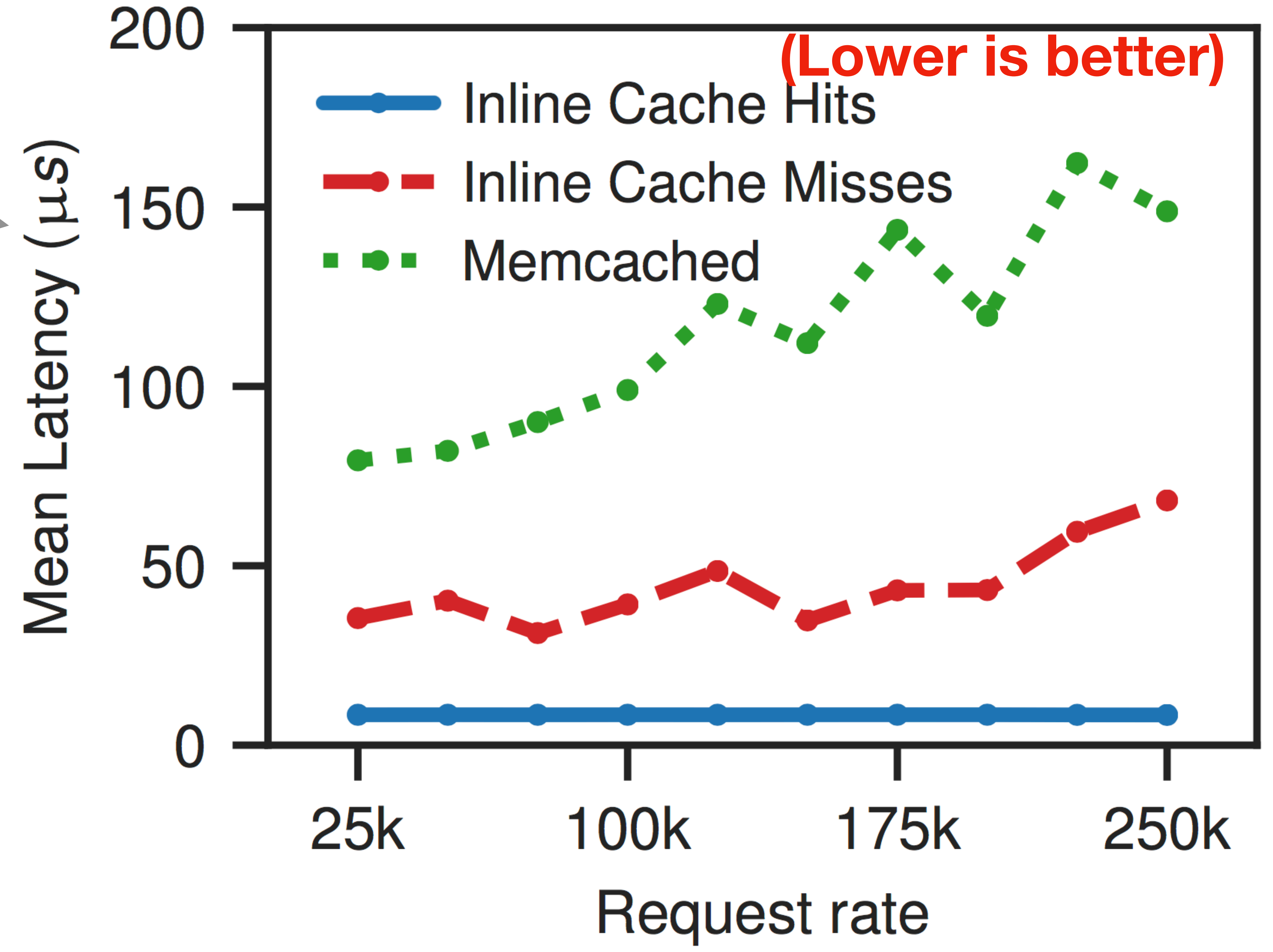
$$\frac{\text{CPU} \quad \text{Rate} < 2 \times 10^8 \quad \text{PacketSize} > 1000}{\text{header\_compress}} \left[\begin{array}{l} \text{Lat.} \mapsto \text{Lat.} + 7.4 \times 10^{-3} \\ \text{Rate} \mapsto \text{Rate} \times \frac{189.9}{194.75} \\ \text{once Power} \mapsto \text{Power} + 150\,\text{W}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

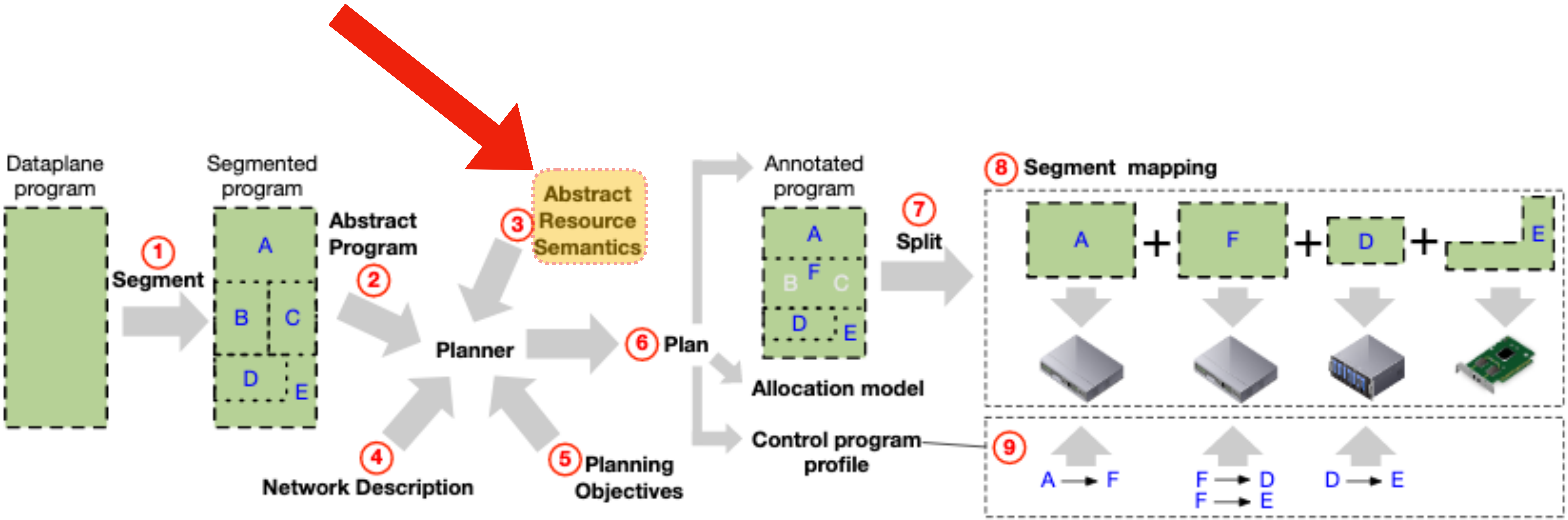
$$\frac{\text{FPGA} \quad \text{Rate} < 9.5 \times 10^9 \quad \text{PacketSize} > 100}{\text{header\_compress}} \left[\begin{array}{l} \text{Lat.} \mapsto \text{Lat.} + 6.44 \times 10^{-6} \\ \text{Rate} \mapsto \text{Rate} \times \frac{9.15}{9.3} \\ \langle\text{LUTs}\rangle \mapsto \text{LUTs} + 24.4\%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

In-Network Program Examples

- Layer 2+ FEC (Forward Error Correction)
- Traffic compression
- In-network caching of key-value requests



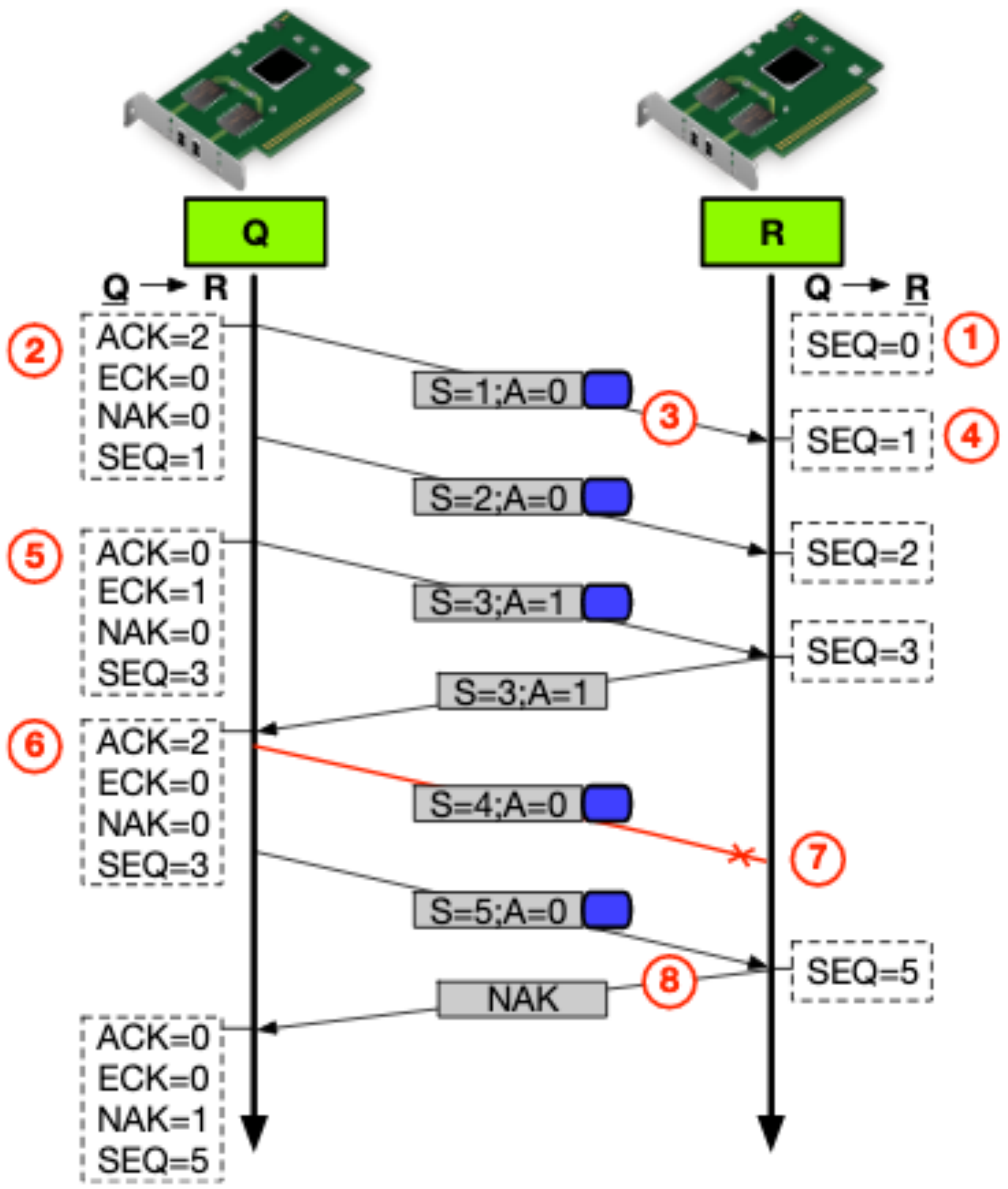




Dataplane program
Segmented program
A
B
C
D
E
1 Segment
Abstract Program
2
3 Abstract Resource Semantics
Planner
4 Network Description
5 Planning Objectives
6 Plan
Annotated program
A
B F C
D
E
7 Split
Allocation model
Control program profile
8 Segment mapping
A
F
D
E
9
A → F
F → D
F → E
D → E

Flightplan

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

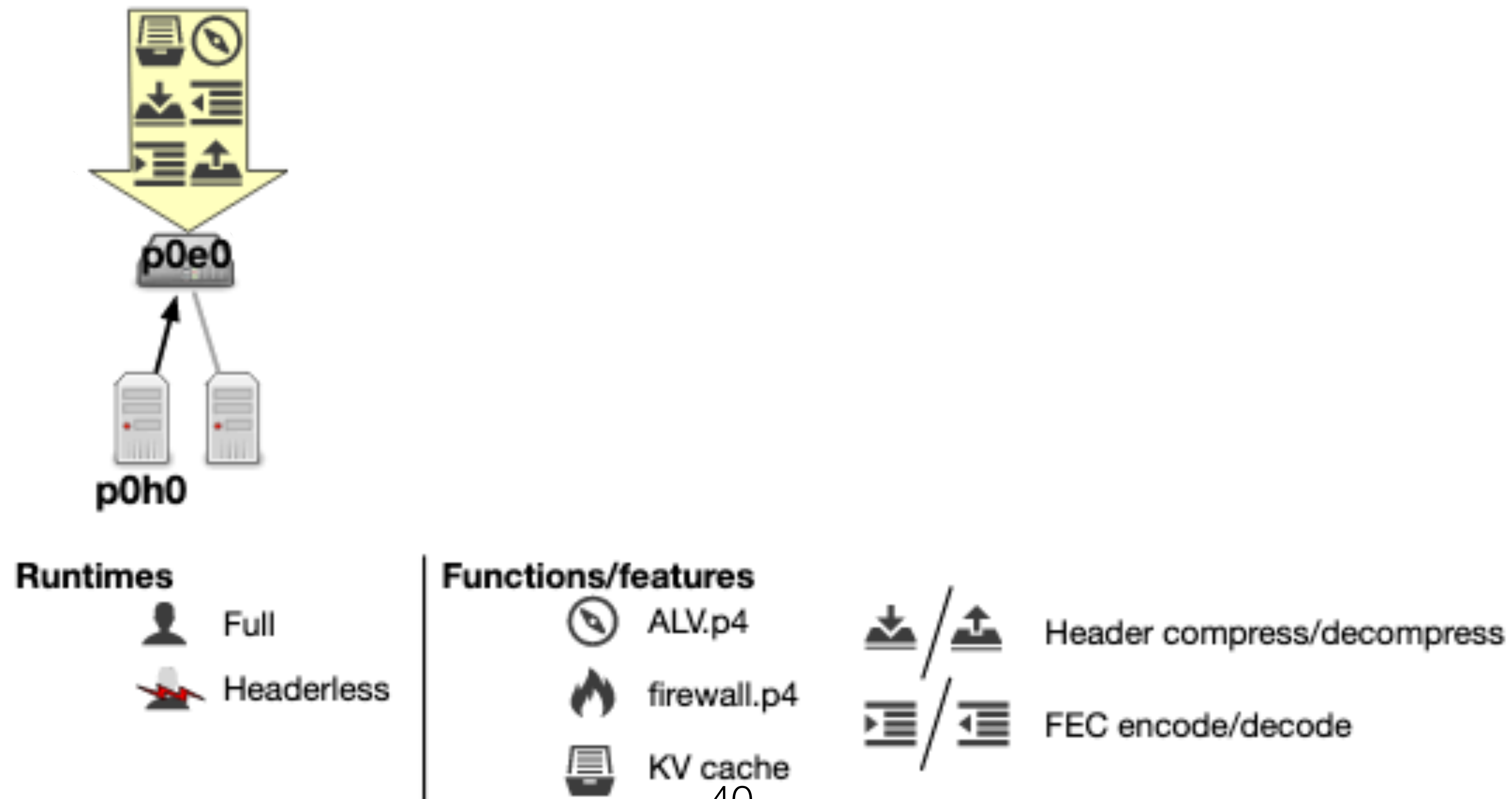


Fig 7: Multiple Programs vs Runtimes vs Splits in same network (Simulation)

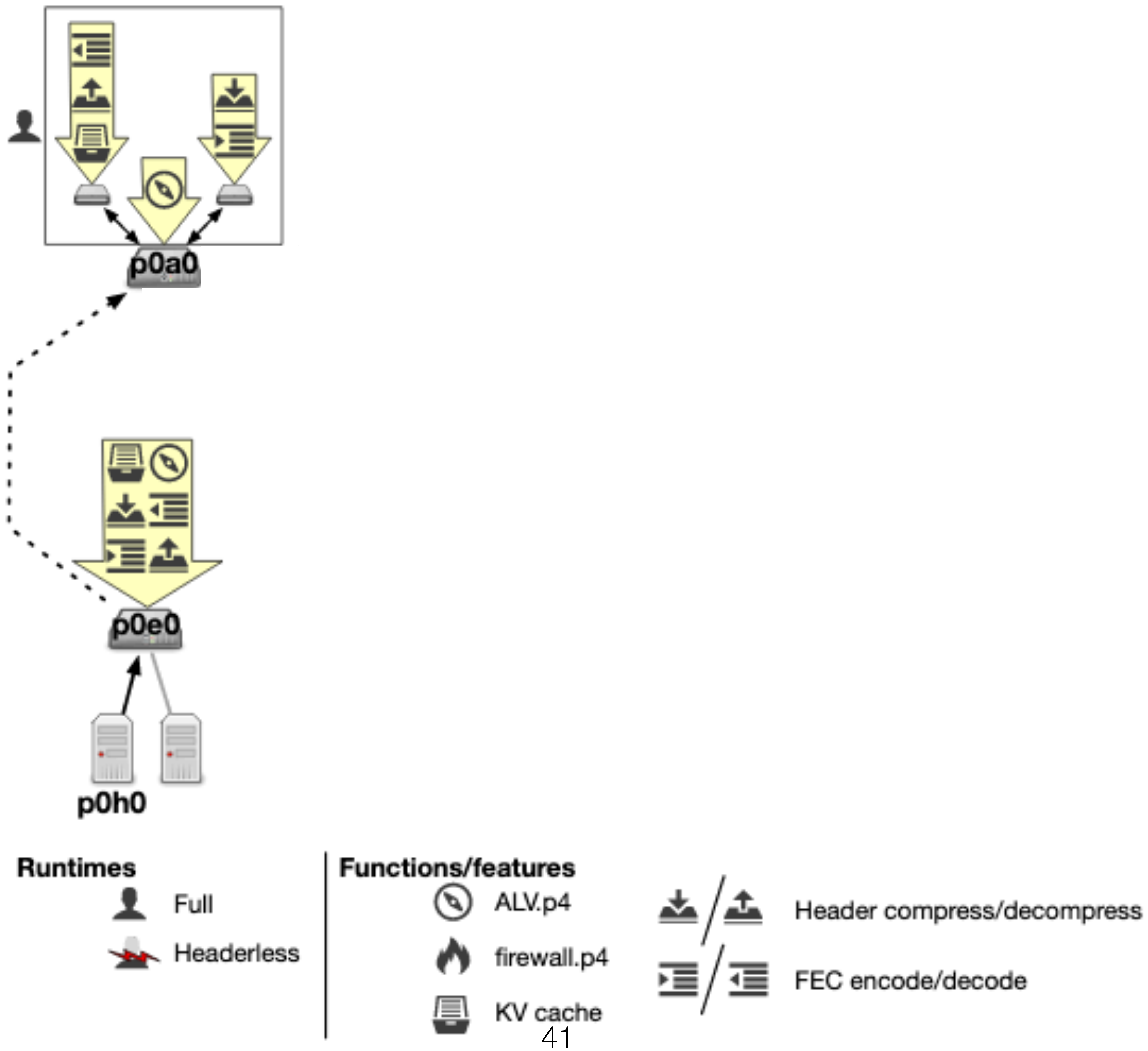
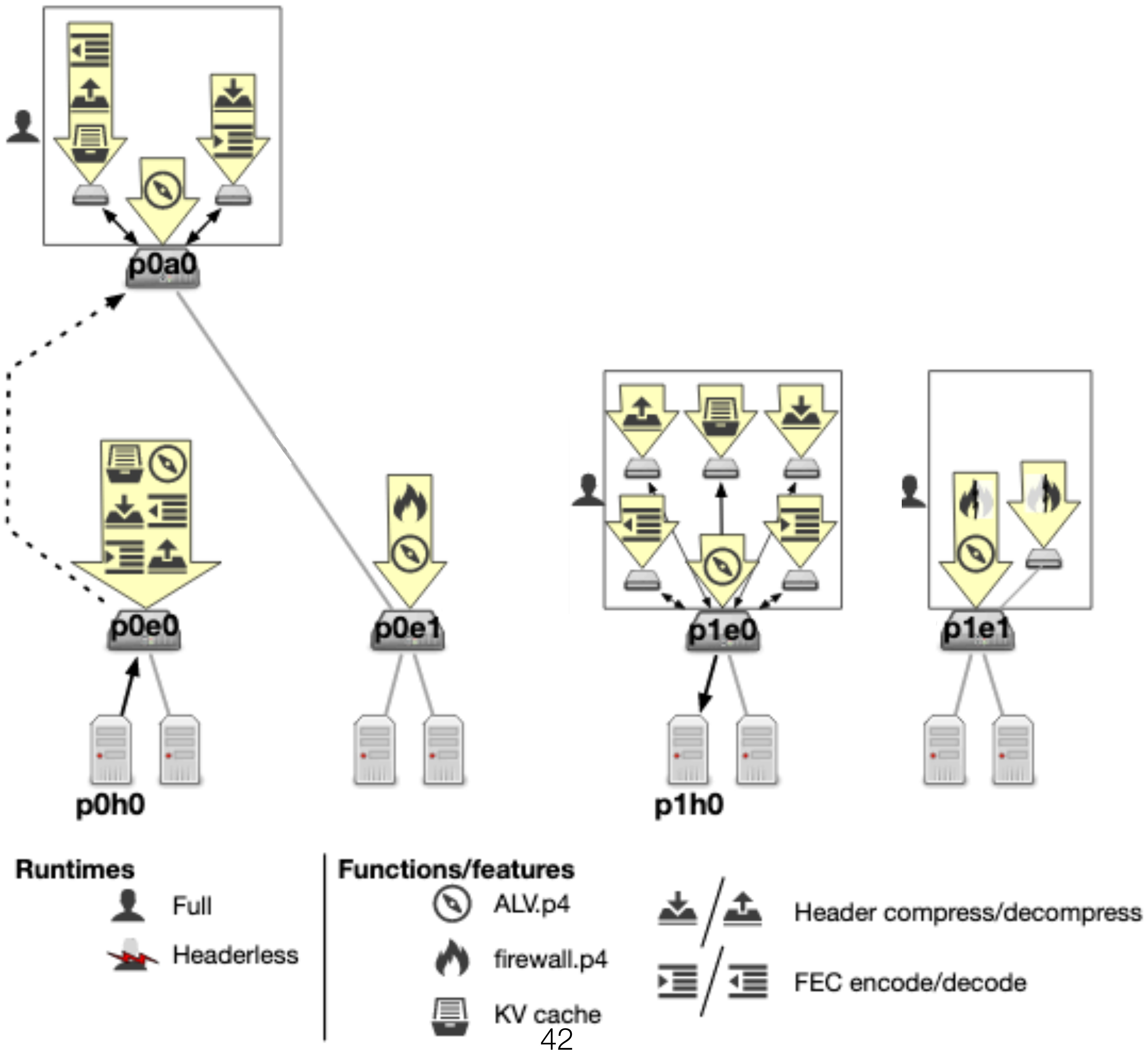
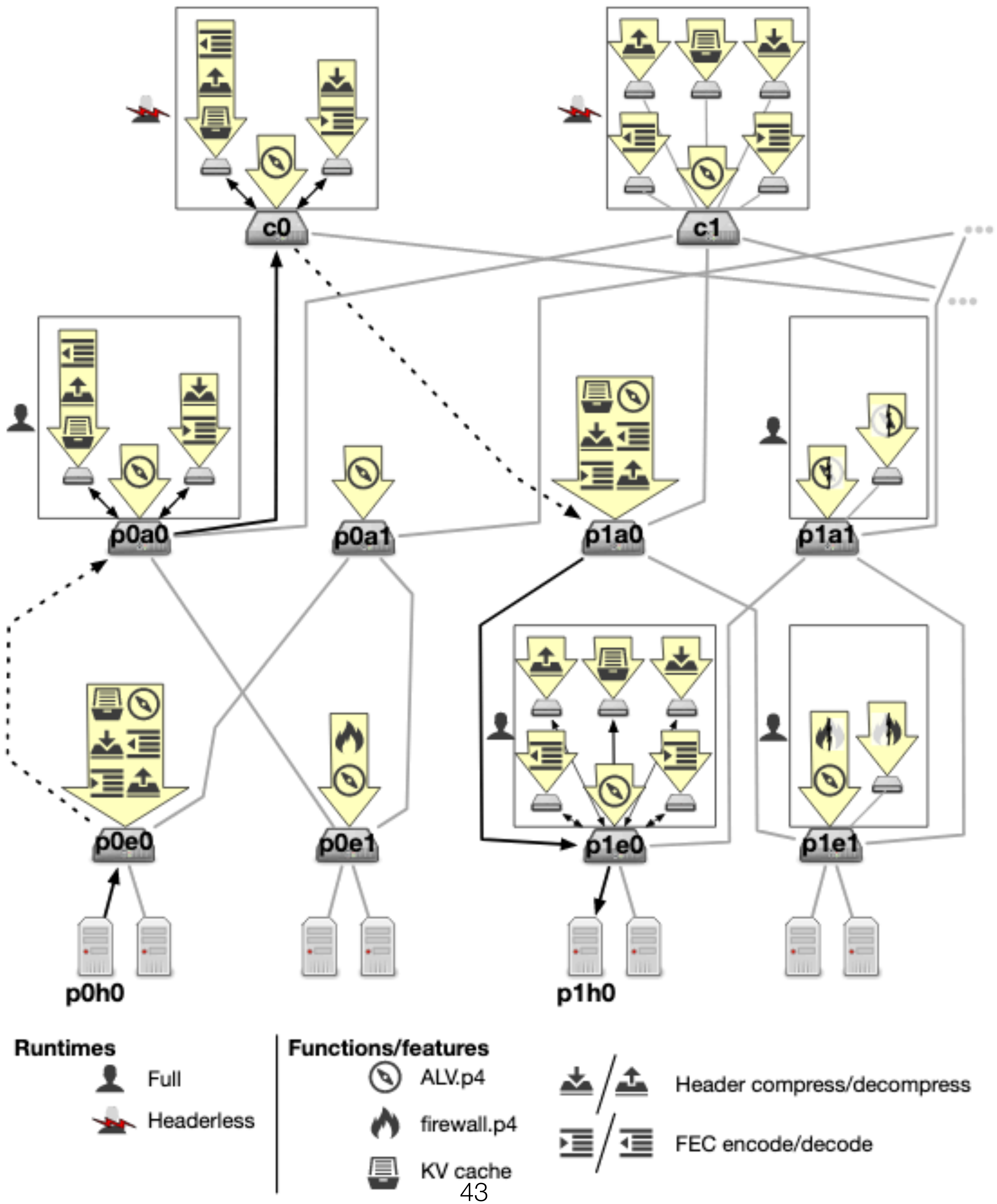


Fig 7: Multiple Programs vs Runtimes vs Splits in same network (Simulation)




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/demo

Ack: Haoxian Chen, Max Demoulin, Joel Hypolite, Pardis Pashakhanloo, Lei Shi, Nishanth Shyamkumar, Caleb Stanford, Ke Zhong