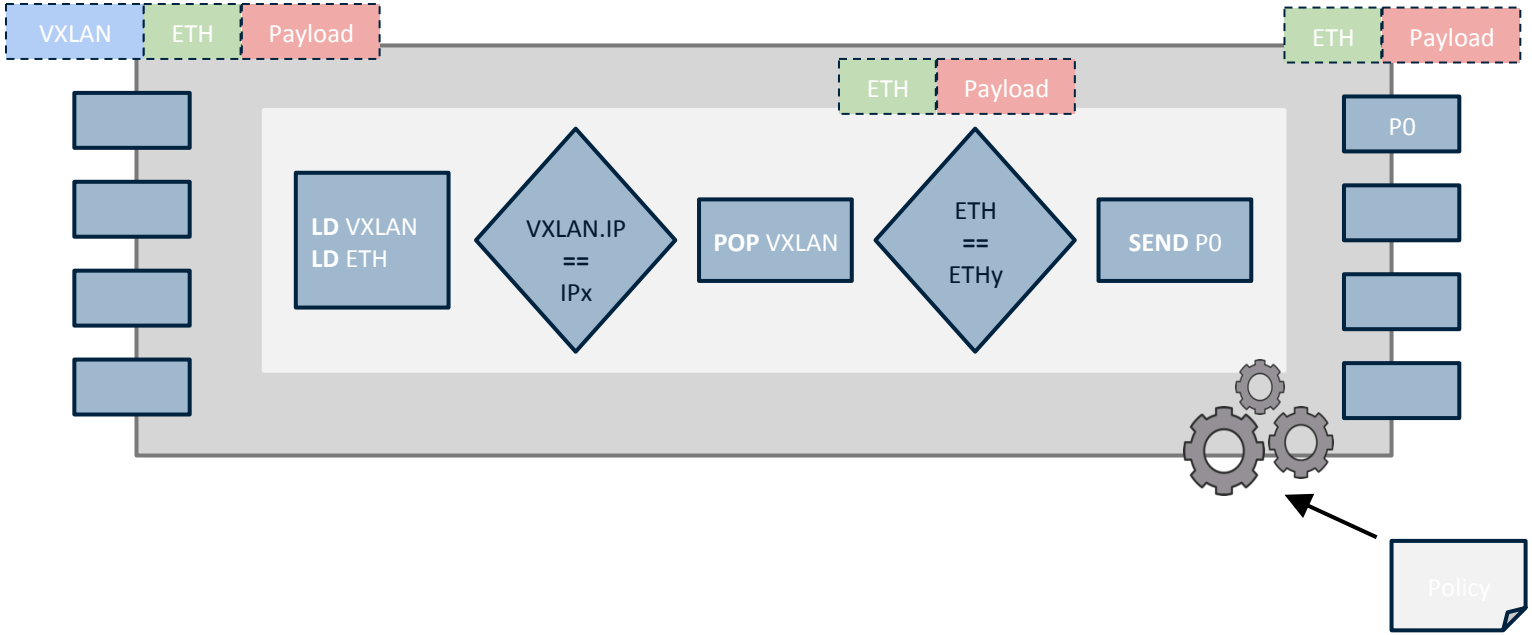
Software Defined Networking

Dr. Nick Feamster
Professor

In this course, you will learn about software defined networking and how it is changing the way communications networks are managed, maintained, and secured.

Need for a Programmable Data Plane

Protocol Independence *(e.g., GRE, VXLAN, BFD)*



Writing Network Programs

- We expect that network programmers will organize their programs into *libraries* and *composable modules*
  - These packet processing modules will likely be reusable
  - They may also be in entirely different high-level languages
- Programmers will use these libraries to write more complex packet processing programs

Programmers need mechanisms for **compiling** these modules to a single hardware target.

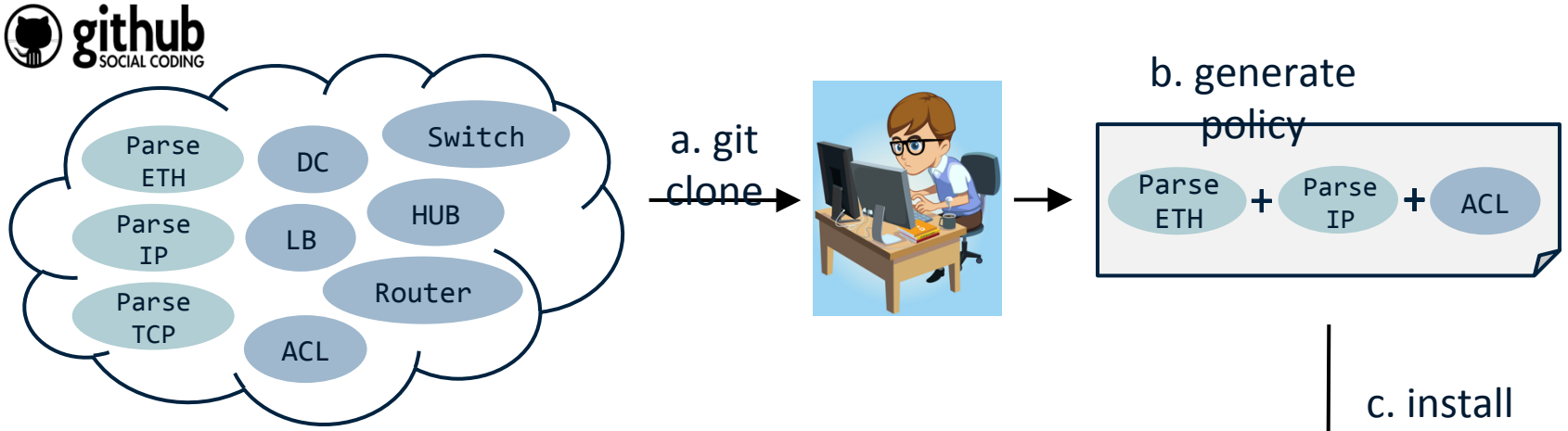
Programming Languages

- Multiple languages tend to make it challenging to achieve direct compilation to a hardware target
  - For example, with languages like C, Java, Python, compiler designers were faced with the issue of how to compile these languages to different targets (e.g., ARM, and Intel's x86)
- Instead of compiling each language directly to a given target, designers developed an intermediate representation (IR)
  - The intermediate language acted as a sweet-spot in dividing the compiler tasks into two phases: **front-** and **back-end**

Goals of an IR

- Must be language- and target-independent
  - Should be expressive enough to be produced by language-specific front-ends, and must be functional enough to produce layouts for a diverse set of hardware targets
- Optimize packet-processing pipelines using both target-specific and target-agnostic optimization techniques
  - e.g., for area, power, or latency
- Optimize the layout of the resulting packet-processing program

Compiling Policy



However, there is an issue here …

- ACL might only be operating on IP src/dst
- In which case, rest of the fields are dead

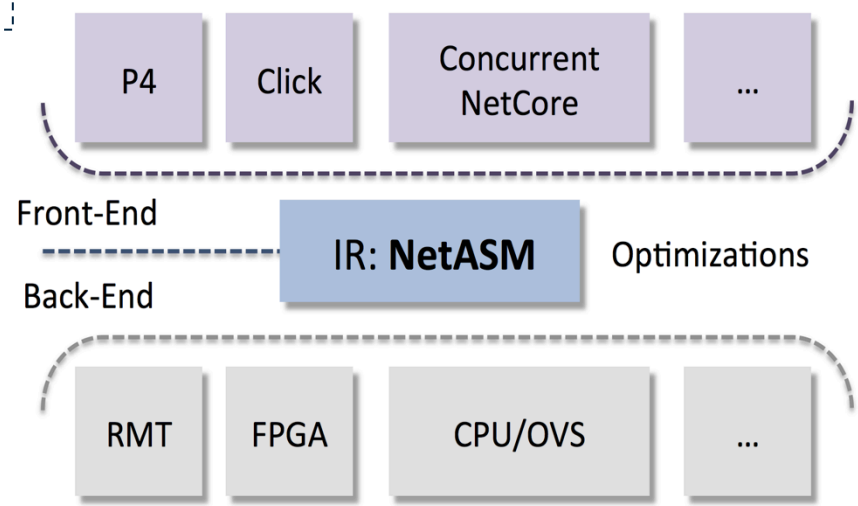
- A **naïve compilation** will add these dead fields in the data plane

NetASM: An Intermediate Representation

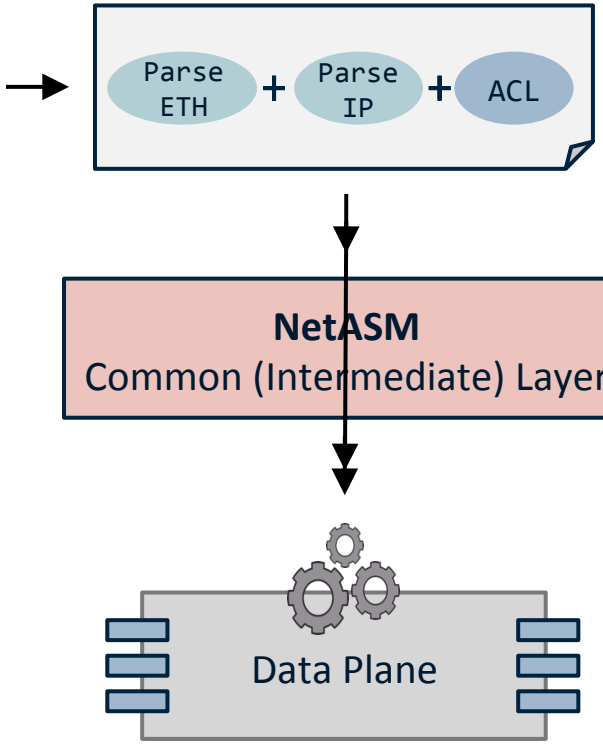
Enables a **common platform** for writing optimizations for programmable data planes

NetASM Design Features

- Provides an abstract data plane model
- Two kinds of state
  a. Per-packet state (e.g., fields)
  b. Persistent state (e.g., tables)
- Modes of execution
  a. Sequential for invoking pipelined parallelism
  b. Concurrent for latency sensitive applications
  c. Atomic for stateful operations using shared state
- 23 primitive instructions



Compiling Policy “Using NetASM”



NetASM acts as a common (intermediate) layer

- Improves quality of code using optimizations like code-motion and dead-code/store elimination
  - Performs conventional data and control flow analyses
  - Uses target agnostic and specific cost model
- Improves for metrics like area, latency, and throughput

NetASM: An IR for SDN Compilers

- Provides an abstract data plane model
- Two kinds of state
  - Per-Packet state (i.e., header fields)
  - Persistent state (i.e., tables)
- Modes of execution
  - Sequential for invoking pipeline parallelism
  - Concurrent for latency sensitive applications
  - Atomic for stateful operations using shared state
- 23 primitive instructions

An Example Program in NetASM

```
# Declare tables
mtable = ([(eth_hdr, 48, Binary)], TBL_SIZE, TableType.CAM)
ptable = ([(outport_bitmap, 2)], TBL_SIZE, TableType.RAM)
```

```
# Parse ETH header
ADD eth_dst, 48
ADD eth_src, 48
ADD eth_type, 16
LD eth_dst, (0, 48)
LD eth_src, (48, 48)
LD eth_type, (96, 16)
```

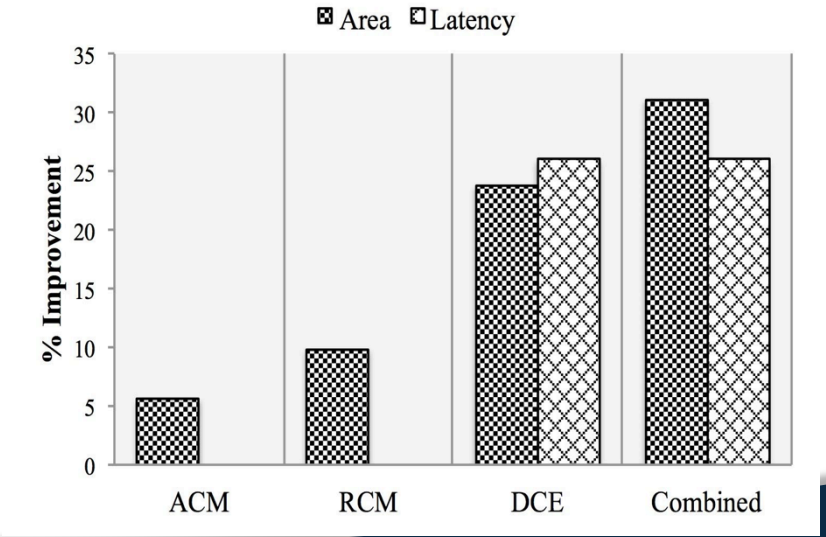
```
# Match-Action Table
ADD index, 16
LKt index, mtable, [eth_src]
BR index, Neq, -1, “LBL_MAT_0”
CTR “MTABLE_MISS”, “” # Case there is no match
JMP “LBL_HLT”
LBL “LBL_MAT_0” # Case there is a match
LDt outport_bitmap, ptable, index
LBL “LBL_HLT”
HLT
```

Program Optimizations

- Target-agnostic optimizations
  - Improving quality of code using optimizations like
    - Code-motion, and dead-code/dead-store elimination
  - Using conventional data and control flow analyses
  - Using target-agnostic cost model
- Target-specific Optimizations (future work)
  - Using target-specific information and cost models
- Improve for metrics like area, latency, and throughput
  - e.g., CPU cycles and memory utilizations for software switches
  - e.g., LUTs, FFs, and clock cycles for FPGA-based switches
  - e.g., No. of stages, table Size, metadata usage for ASIC/RMT switches

Preliminary Evaluation

- ACL-IPv4 benchmark from ONF Github repository
  https://github.com/NetASM/ACL-IPv4-Example
- Results using our abstract cost model (details in the paper)



Summary and Future Work

- NetASM: A Common IR for Programmable Data Planes
  - Enables a compiler to optimize a high-level packet processing program for a diversity of targets
  - Uses a target-independent machine model and cost semantics to optimize the program
  - Leads to better architectural explorations

- Next Steps
  - Complete the language specification for NetASM
  - Explore opportunities for optimizations that can be applied across different classes of network device architectures