Georgia Institute of Technology

Dr. Nick Feamster
Associate Professor

Software Defined Networking

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

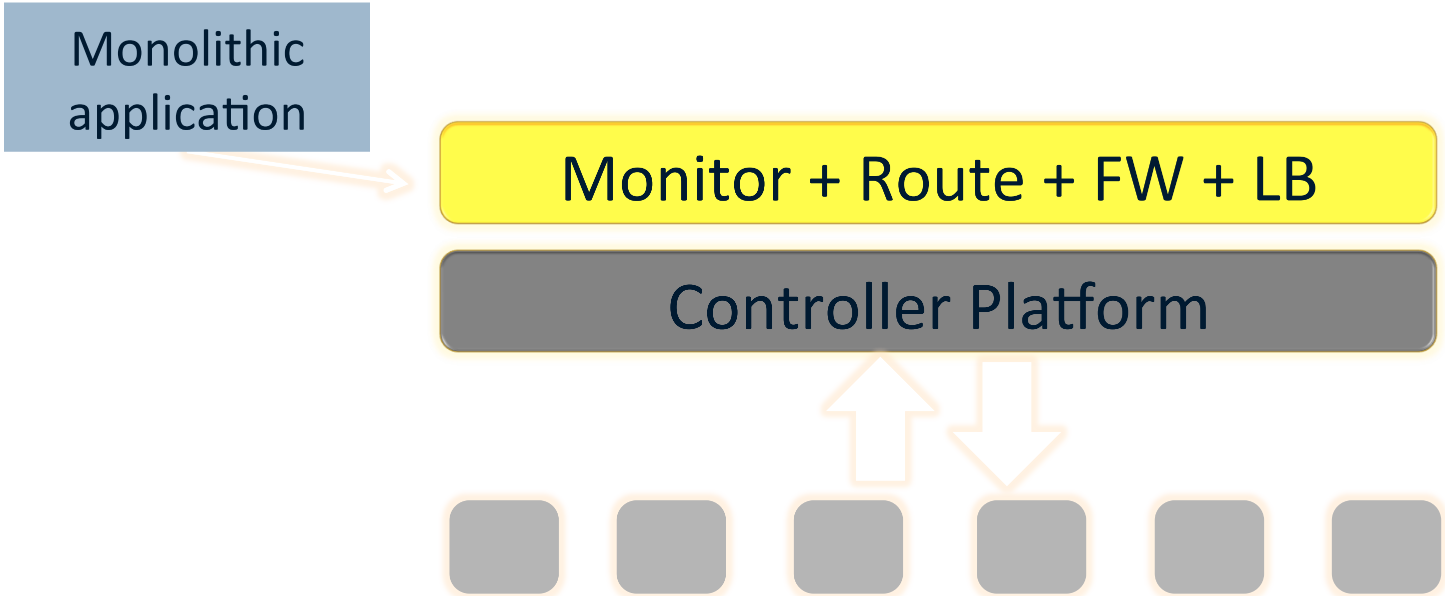
School of Computer Science

Module 6.3: Programming SDNs

- Four Lessons
  - Motivation for Programming SDNs
  - Programming Languages for SDNs
  - **Composing SDN Control**
  - Event-Driven SDN
- Programming Assignment
- Quiz

Some slides Courtesy of Jennifer Rexford (used with permission)

Networks Perform Many Tasks



Monolithic application

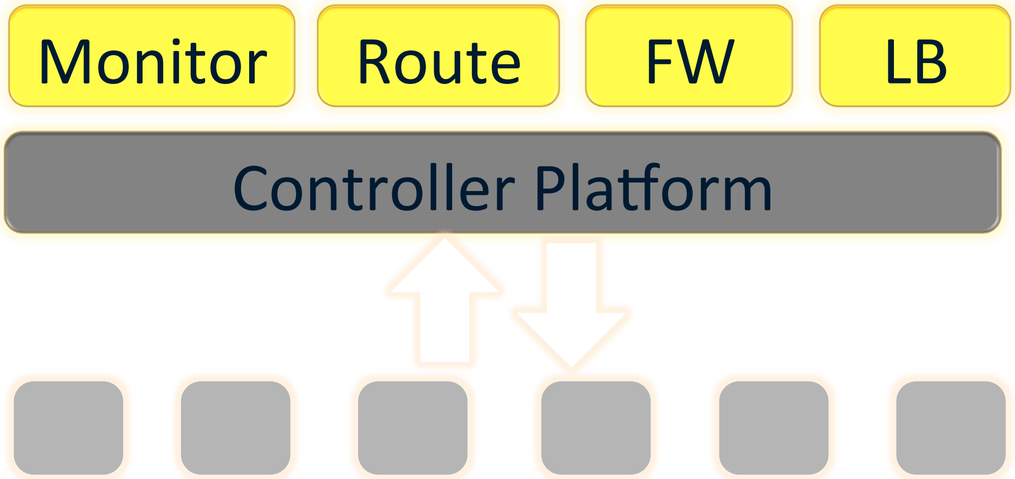
Monitor + Route + FW + LB

Controller Platform

Hard to program, test, debug, reuse, port, …

Solution: Modularize Control

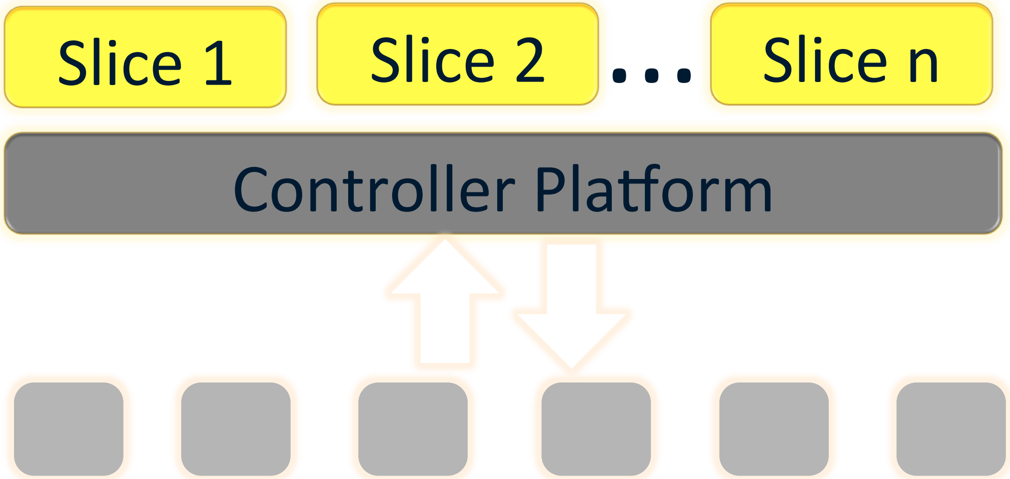
A module for each task



Easier to program, test, and debug
Greater reusability and portability

Modules Are Not Just "Tenants"

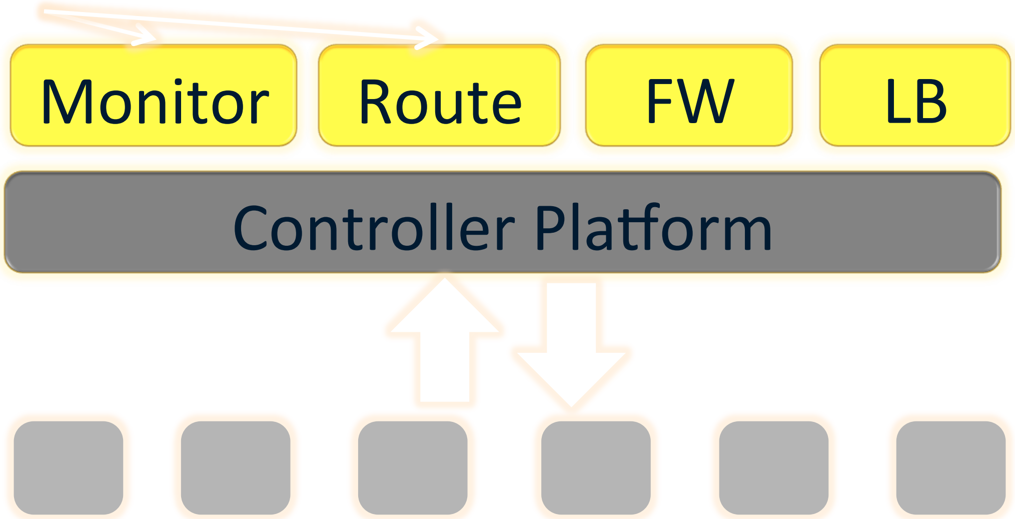
Each module controls a *different* portion of the traffic



Relatively easy to partition *rule space*, *link bandwidth*, and *network events* across modules

Modules Affect the *Same* Traffic

Each module *partially* specifies the handling of the traffic



How to combine modules into a complete application?

Approach: Composition

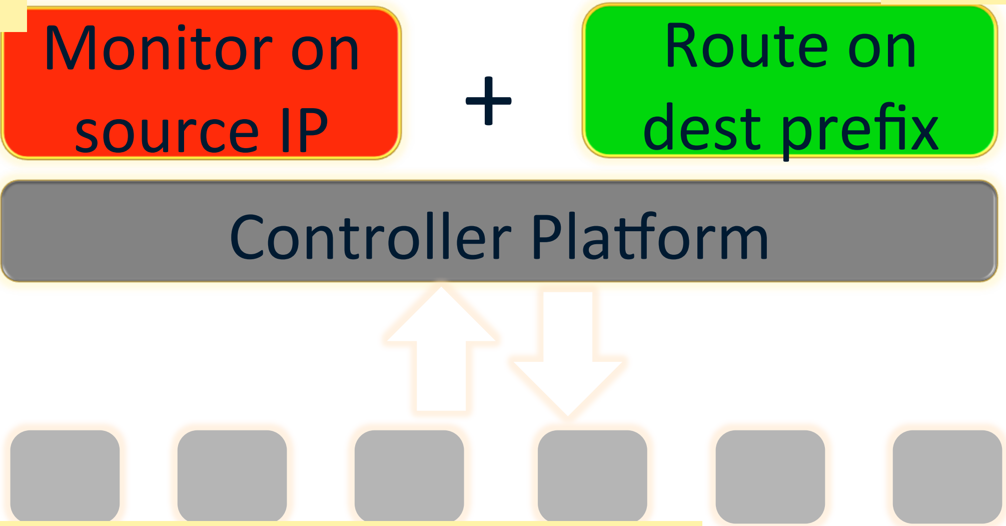
- **Parallel composition:** Perform both operations simultaneously (e.g., counting, forwarding)

- **Sequential composition:** Perform one operation, then the next (e.g., firewall then switch)

Parallel Composition

srcip = 5.6.7.8 → count
srcip = 5.6.7.9 → count

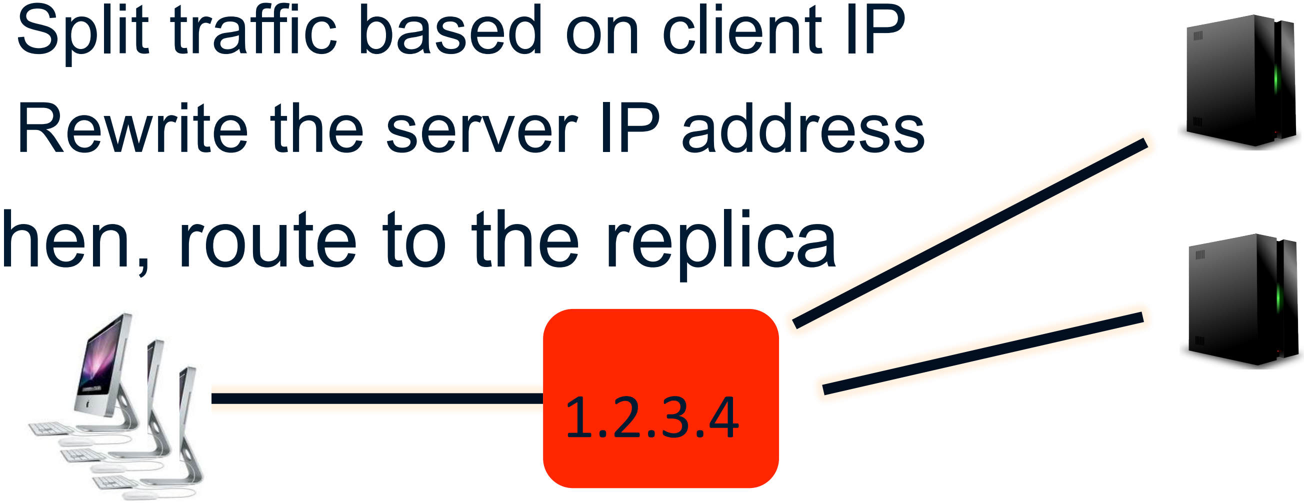
dstip = 1.2/16 → fwd(1)
dstip = 3.4.5/24 → fwd(2)



srcip = 5.6.7.8, dstip = 1.2/16 → fwd(1), count
srcip = 5.6.7.8, dstip = 3.4.5/24 → fwd(2), count
srcip = 5.6.7.9, dstip = 1.2/16 → fwd(1), count
srcip = 5.6.7.9, dstip = 3.4.5/24 → fwd(2), count

Another Example: Server Load Balancer

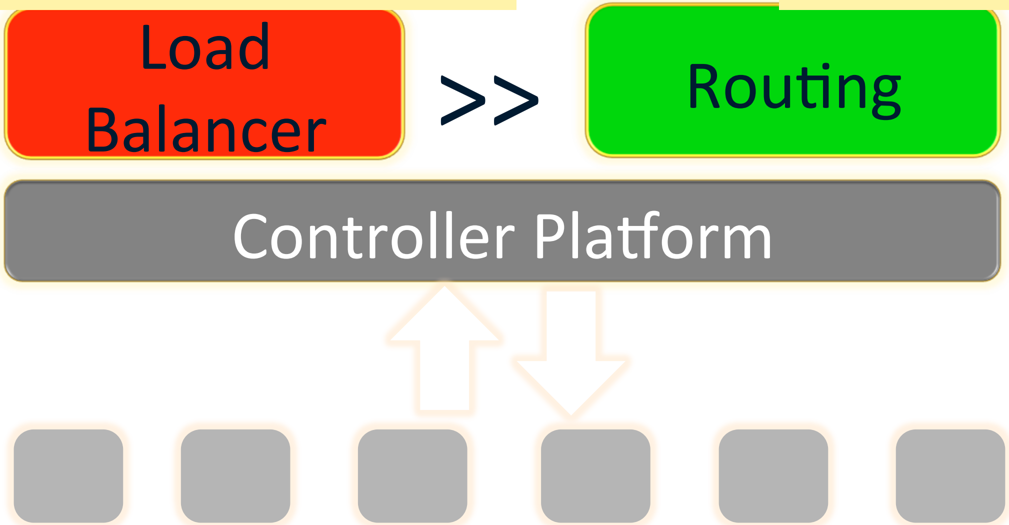
- Spread client traffic over server replicas
  - Public IP address for the service
  - Split traffic based on client IP
  - Rewrite the server IP address
- Then, route to the replica



Sequential Composition

srcip = 0*, dstip=1.2.3.4 → dstip=10.0.0.1
srcip = 1*, dstip=1.2.3.4 → dstip=10.0.0.2

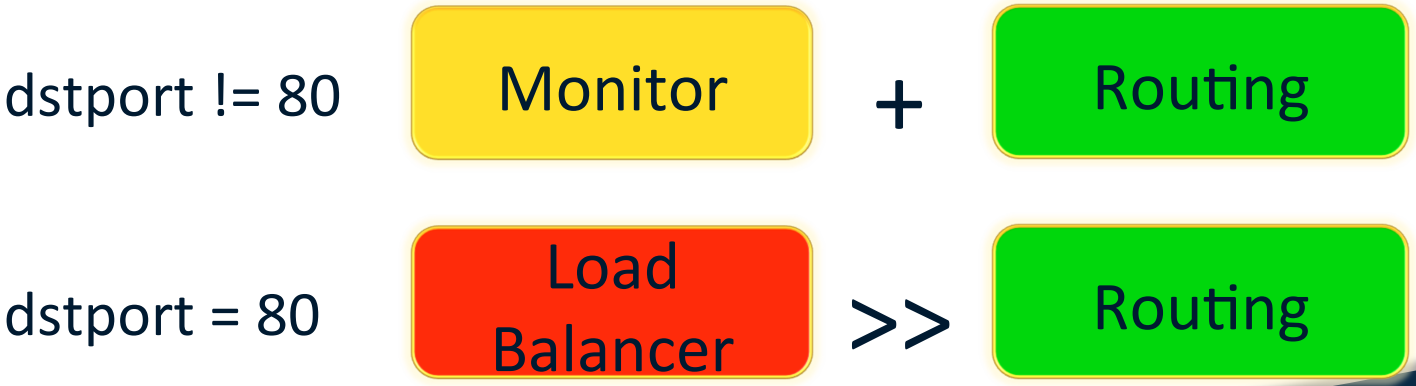
dstip = 10.0.0.1 → fwd(1)
dstip = 10.0.0.2 → fwd(2)



srcip = 0*, dstip = 1.2.3.4 → dstip = 10.0.0.1, fwd(1)
srcip = 1*, dstip = 1.2.3.4 → dstip = 10.0.0.2, fwd(2)

Dividing the Traffic Over Modules

- Predicates
  - Specify which traffic traverses which modules
  - Based on input port and packet-header fields

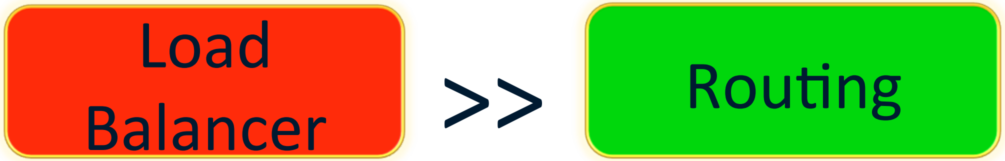


dstport != 80 Monitor + Routing

dstport = 80 Load Balancer >> Routing

Partially Specifying Functionality

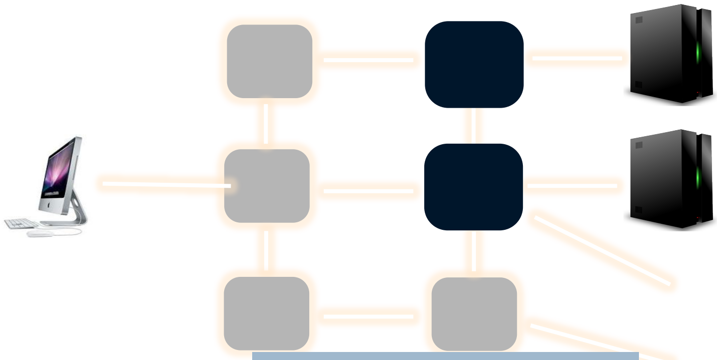
- A module should not specify *everything*
  - Leave some flexibility to other modules
  - Avoid tying the module to a specific setting
- Example: load balancer plus routing
  - Load balancer spreads traffic over replicas
  - … without regard to the network paths



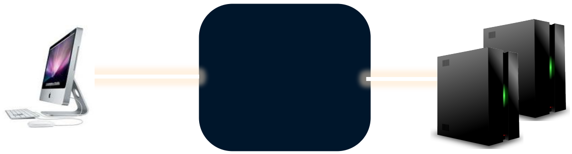
Avoid custom interfaces between the modules

Abstract Topology Views

- Present abstract topology to the module
  - Implicitly encodes the constraints
  - Looks just like a normal network
  - Prevents the module from overstepping



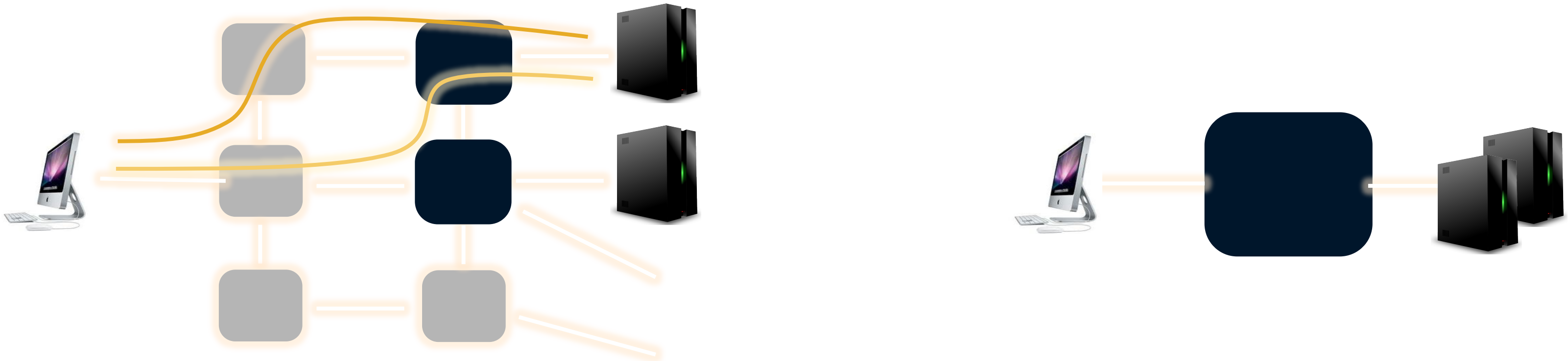
Real network



Abstract view

Separation of Concerns

- Hide irrelevant details
  - Load balancer doesn’t see the internal topology or any routing changes



Summary

- SDN control programs perform many tasks on the same traffic
- Requirements
  - **Compositional operators:** Specifying how to compose those policies
  - **Logical switch abstraction:** Hiding irrelevant details
- **Next Lecture:** Pyretic Language