Software Defined Networking

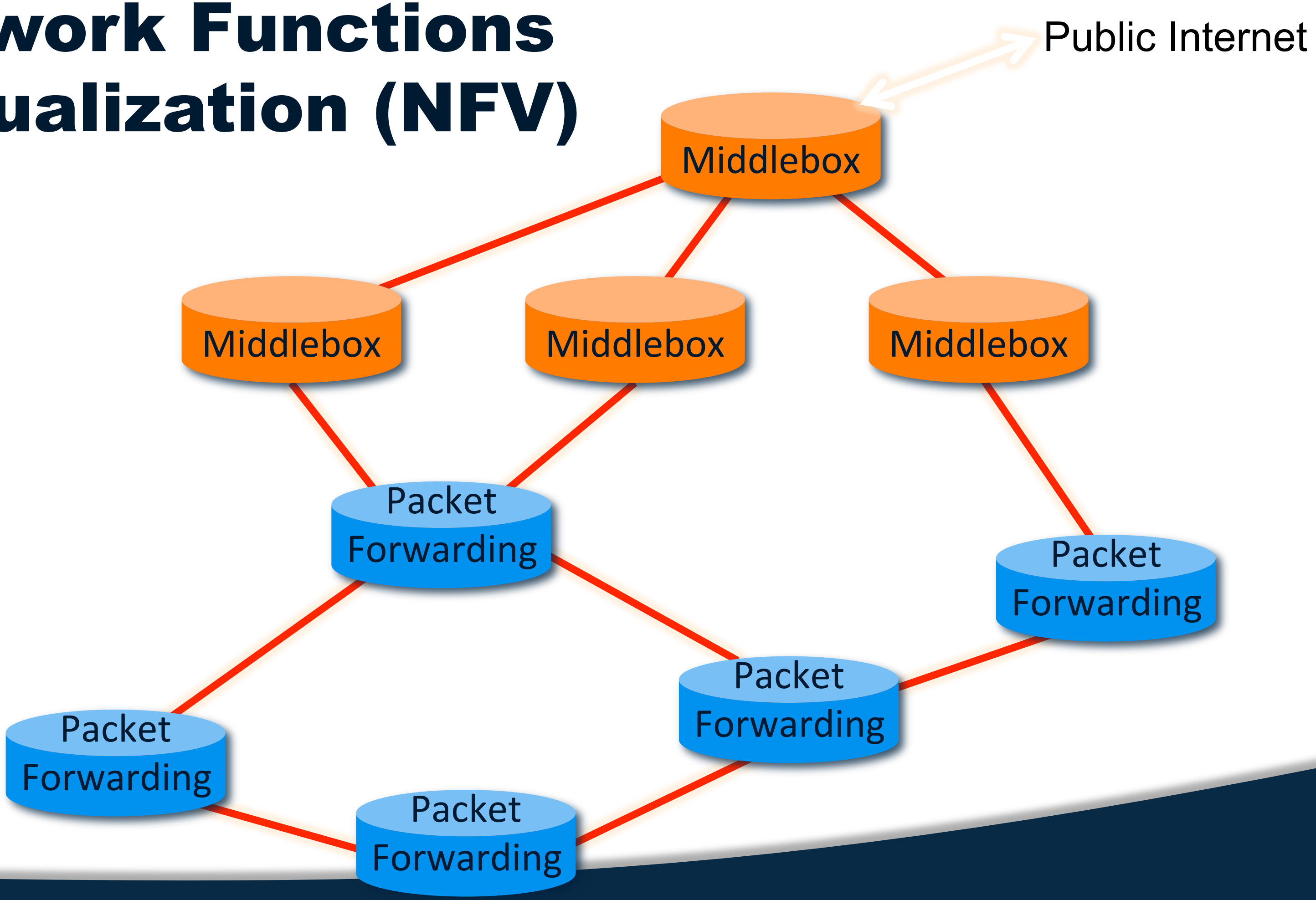
Dr. Nick Feamster
Professor

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

Network Functions Virtualization

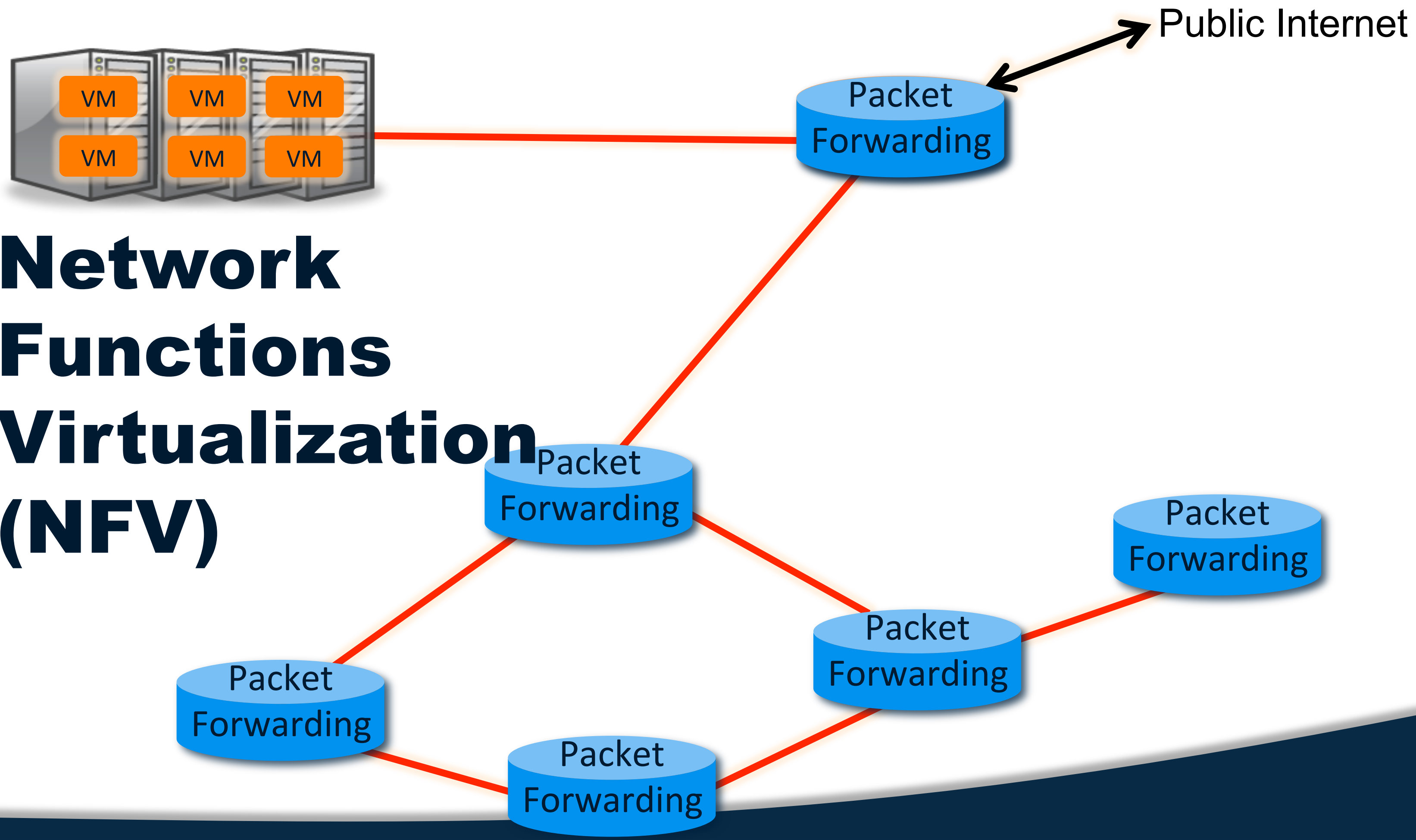
- Overview
- Applications
- Problems in NFV
- One Architecture: Slick

Network Functions Virtualization (NFV)



Software Defined Networking

Network Functions Virtualization (NFV)



Software Defined Networking

Benefits

- Reduced Capex
- Reduced time to market
- Elastic scaling
- Targeted services
- Vendor neutrality

New Use Cases

- Virtualized services for enterprises
- Virtual CDNs
- Virtualized mobile core networks
- Integrate production/testing

Functional Elements, not Middleboxes

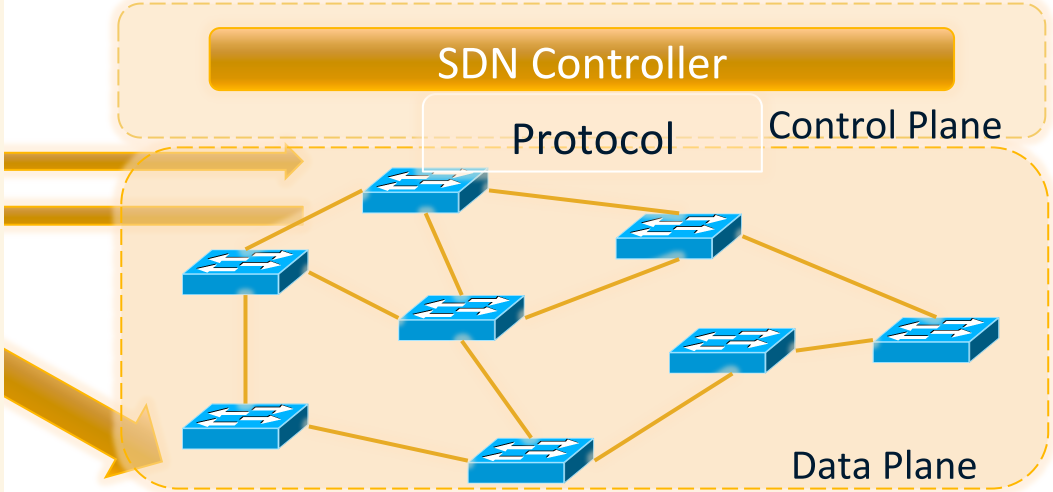
- **WAN Optimizer** = Caching+ Deduplication + Compression+ Encryption+ Forward Error Correction+ Rate Limiter
- **Application Firewall** = IP Defragmenter + Application Detection Engine+ Logger+ Blocker
- **Snort** = IP Defragmenter + Preprocessing + Misuse Detection Engine + Logger

Orchestration and Customizability

- Enable network operators to implement **modular network functions (elements)** without worrying about ***"how"*** and ***"where"*** to install network functions.
- Add **custom middlebox functions** inside network data plane
  - Log flow information for all TCP packets from specific port number.
  - Perform ***f(x)*** on single byte in each packet of a flow space at fixed offset of packet header

Slick Approach

- **Small *functional units* *("elements")*** that can be **reused** and ***composed*** to create more ***complex functions***
- **Programming Abstraction** to add functions inside the network
- **Runtime System** to support the abstraction

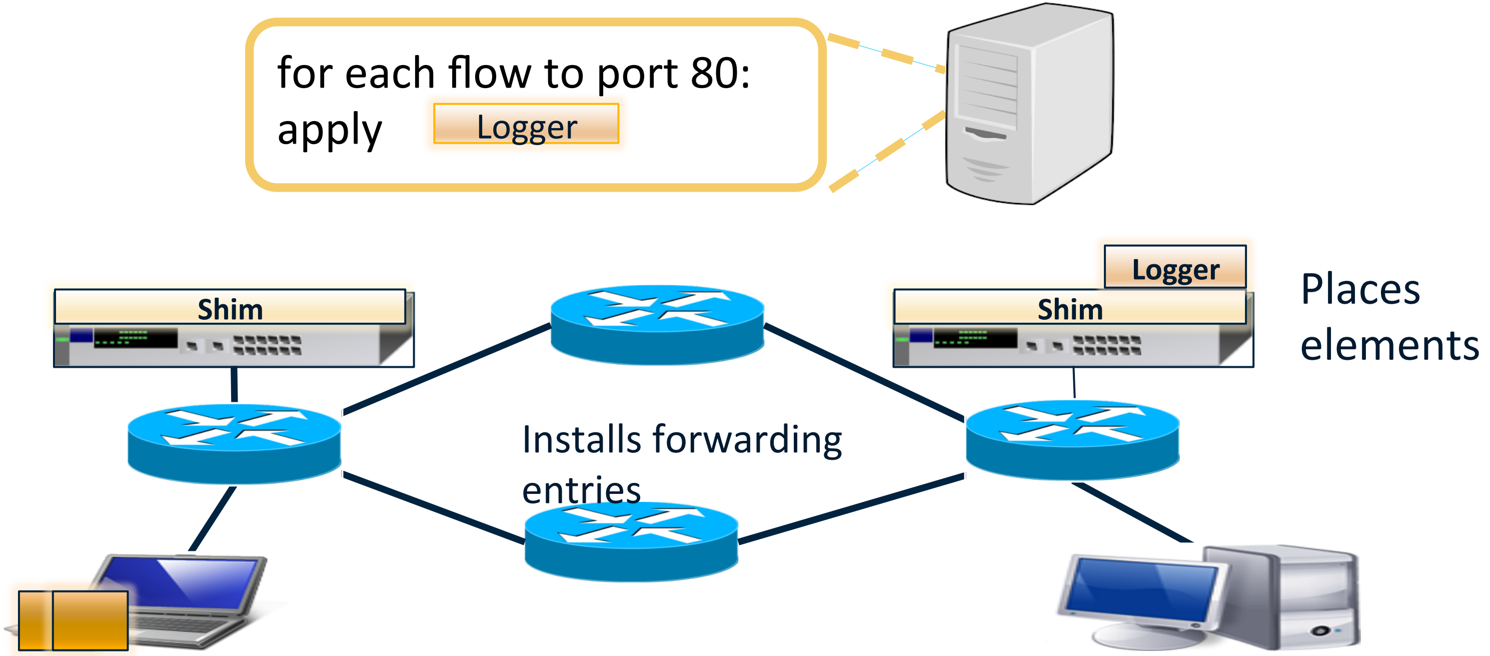


Programming Slick Network Functions

Bilal Anwer (Georgia Institute of Technology), Theophilus Benson (Duke University), Nick Feamster (Princeton University), and Dave Levin (University of Maryland)

ACM SIGCOMM Symposium on SDN Research (SOSR), June 2015.

Programming Abstraction



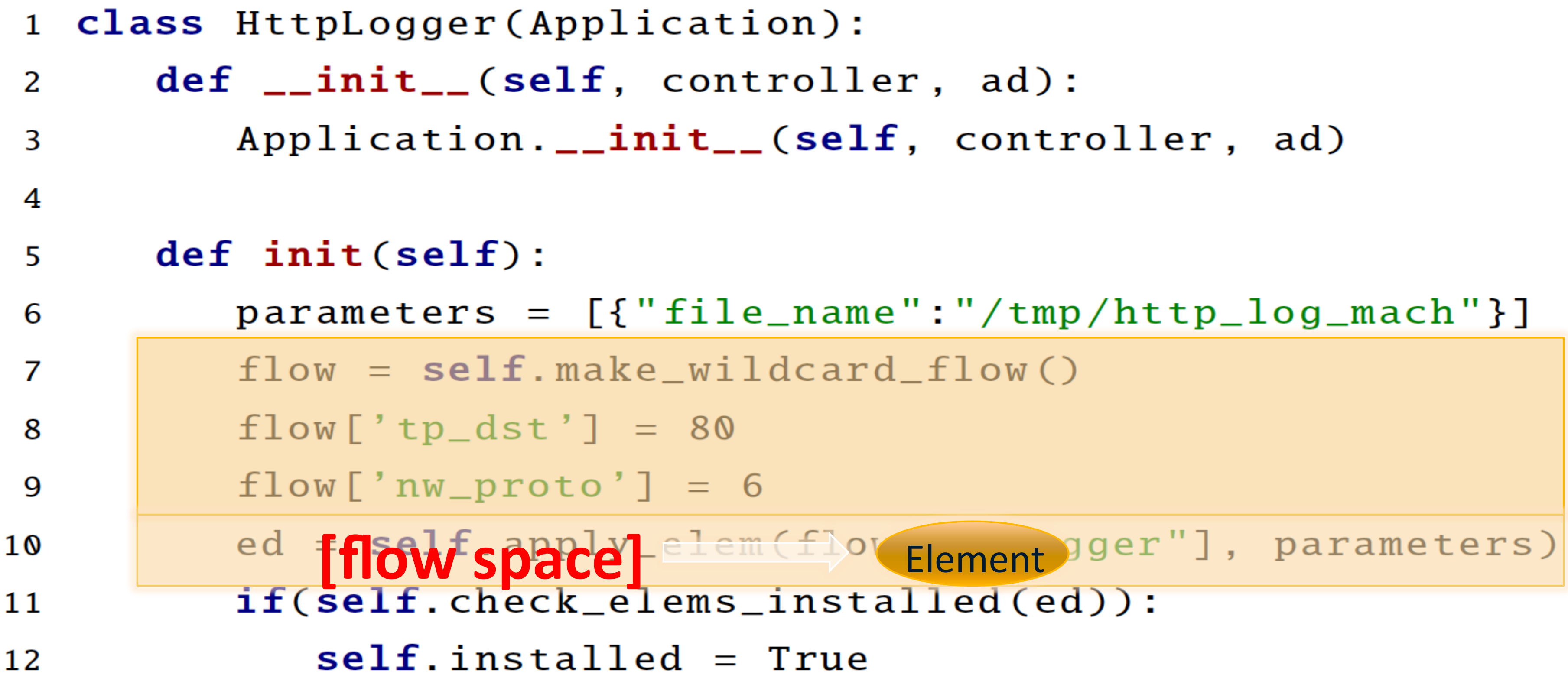
for each flow to port 80:
apply Logger

Places elements

Installs forwarding entries

Software Defined Networking

Slick Application



```python
class HttpLogger(Application):
    def __init__(self, controller, ad):
        Application.__init__(self, controller, ad)

    def init(self):
        parameters = [{"file_name":"/tmp/http_log_mach"}]
        flow = self.make_wildcard_flow()
        flow['tp_dst'] = 80
        flow['nw_proto'] = 6
        ed = self.apply_elem(flow, ["HttpLogger"], parameters)
        if(self.check_elems_installed(ed)):
            self.installed = True
```

[flow space]

Element

Slick Elements

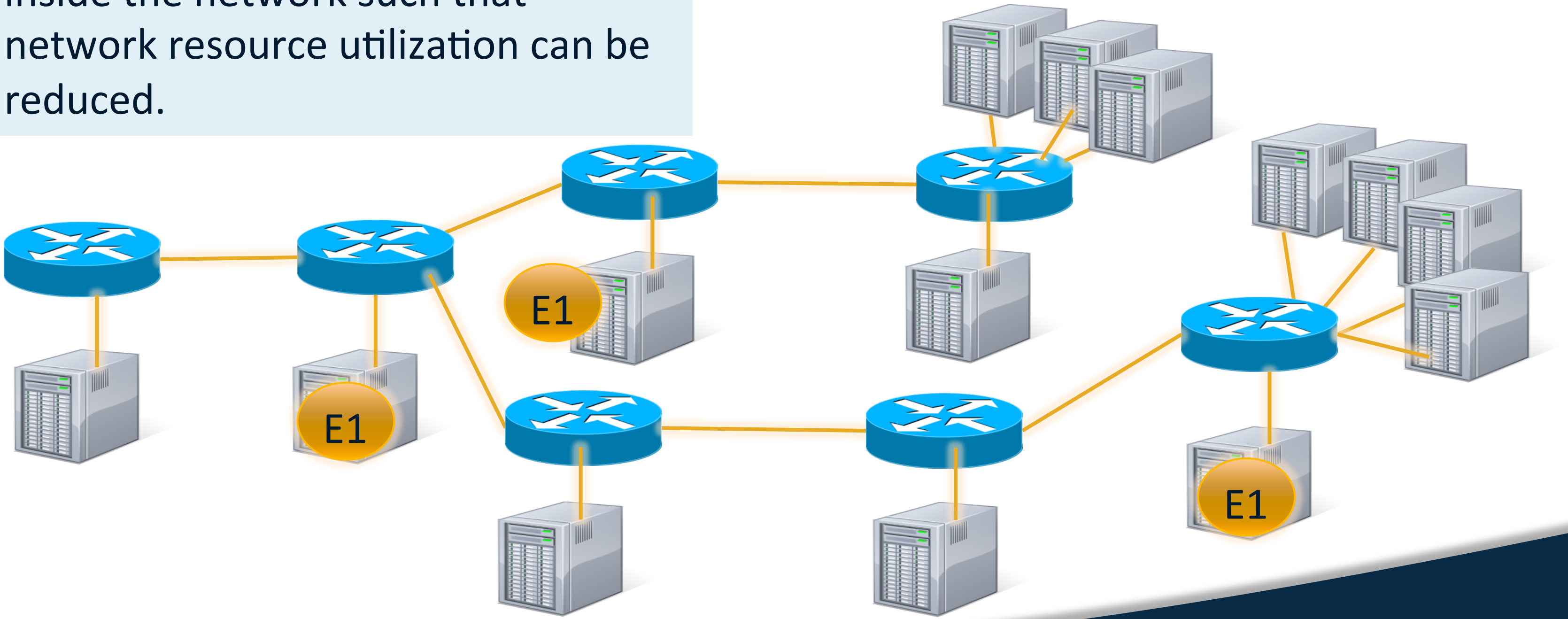
- Basic functional unit in Slick environment
- Inspired by Click elements
- Uniform interface to write functions
- Specification file to describe element properties to Slick Controller

Runtime: Placement

Flow space → E1

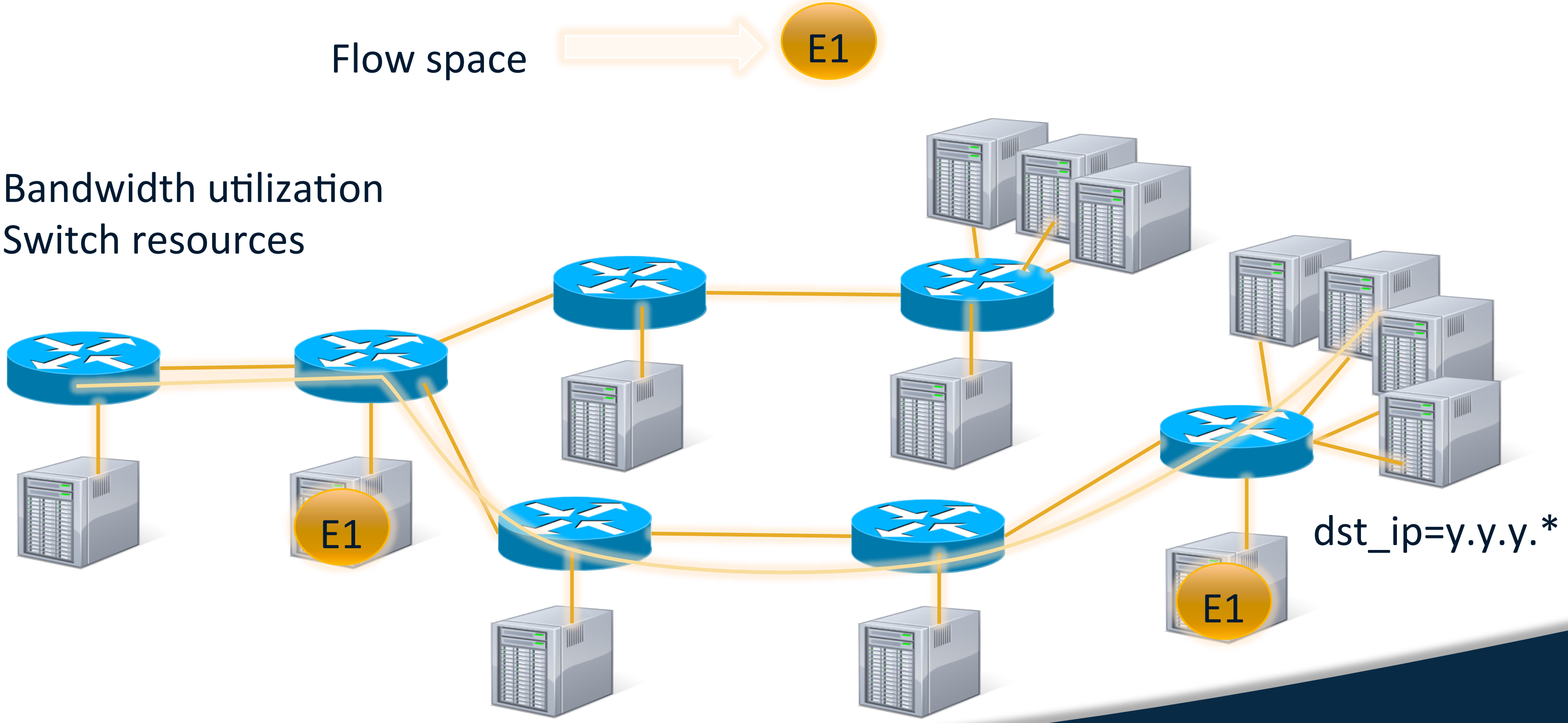
Problem: Given flow space to element mapping, place elements inside the network such that network resource utilization can be reduced.

dst_ip=y.y.y.*

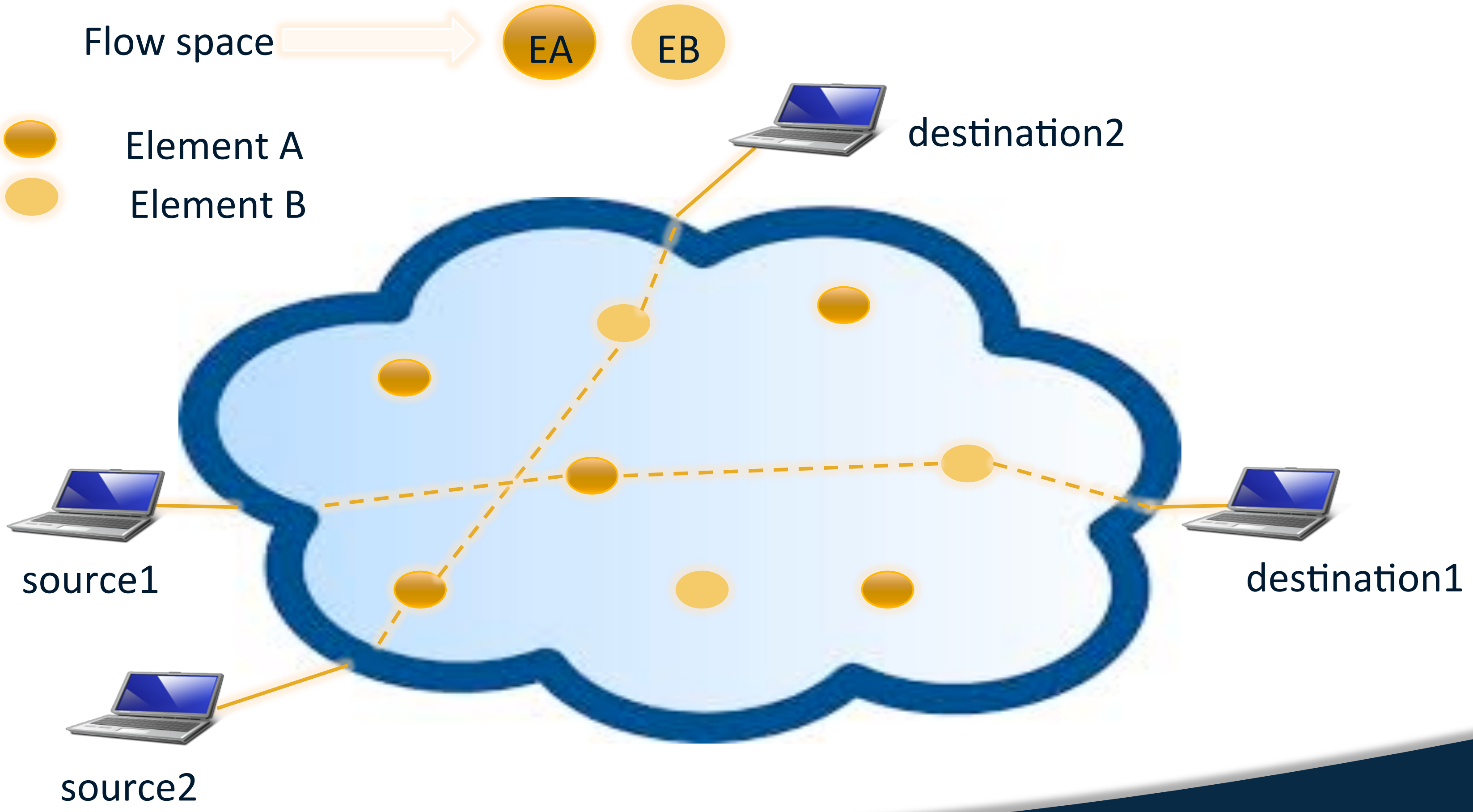


Inflation-Aware Placement

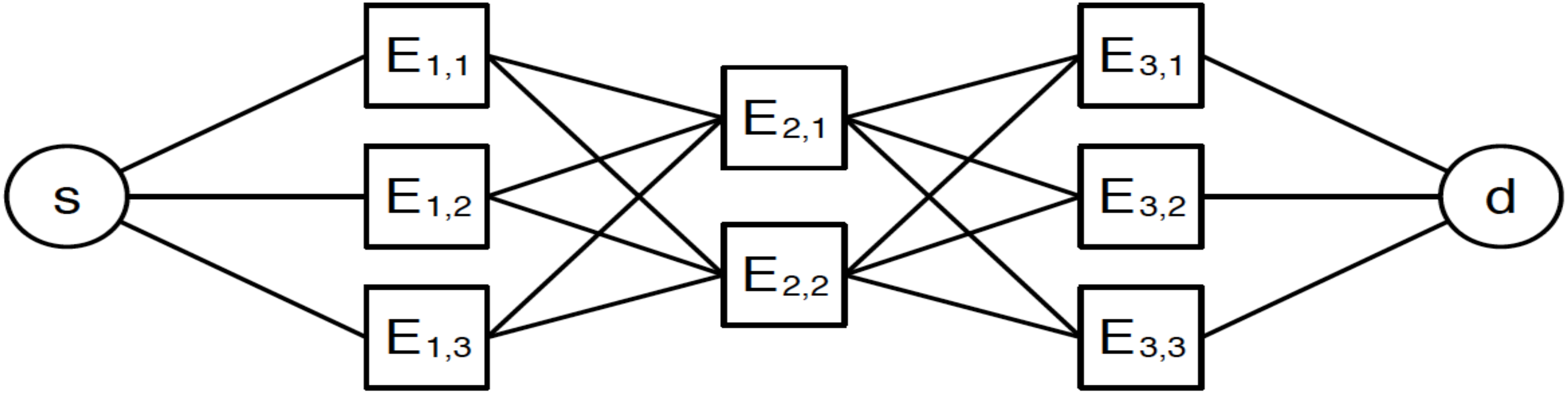
- Bandwidth utilization
- Switch resources



Runtime: Steering

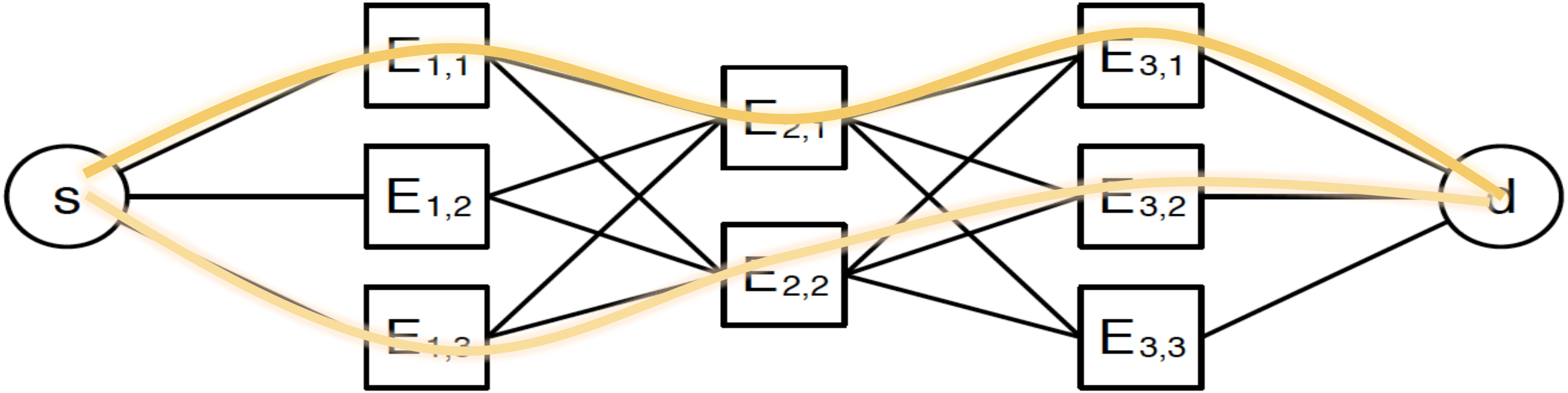


Steering Overlay Graph



- Separately maintained for each flow space to element mapping

Steering Overlay Graph



- Different Path for each source destination pair

Future Work

- Better algorithms for placement and steering
- High speed data plane implementation
  - Isolation, Speed, Instantiation, Migration
- Policy conflicts
  - One flow space dictated by two policies
- Verification of Slick System properties

Conclusion

- NFV: New approach for deploying middlebox functions on virtual appliances in the network
- Slick-based approach for NFV can result in evolvable data plane designs