Practical Foundations for Software-Defined Network Optimization

CCF-1535917, 1536002
https://users.ece.cmu.edu/~vsekar/aitf_sol.html

Anupam Gupta
Carnegie Mellon University

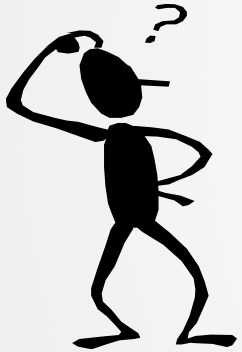
Michael K. Reiter
UNC Chapel Hill

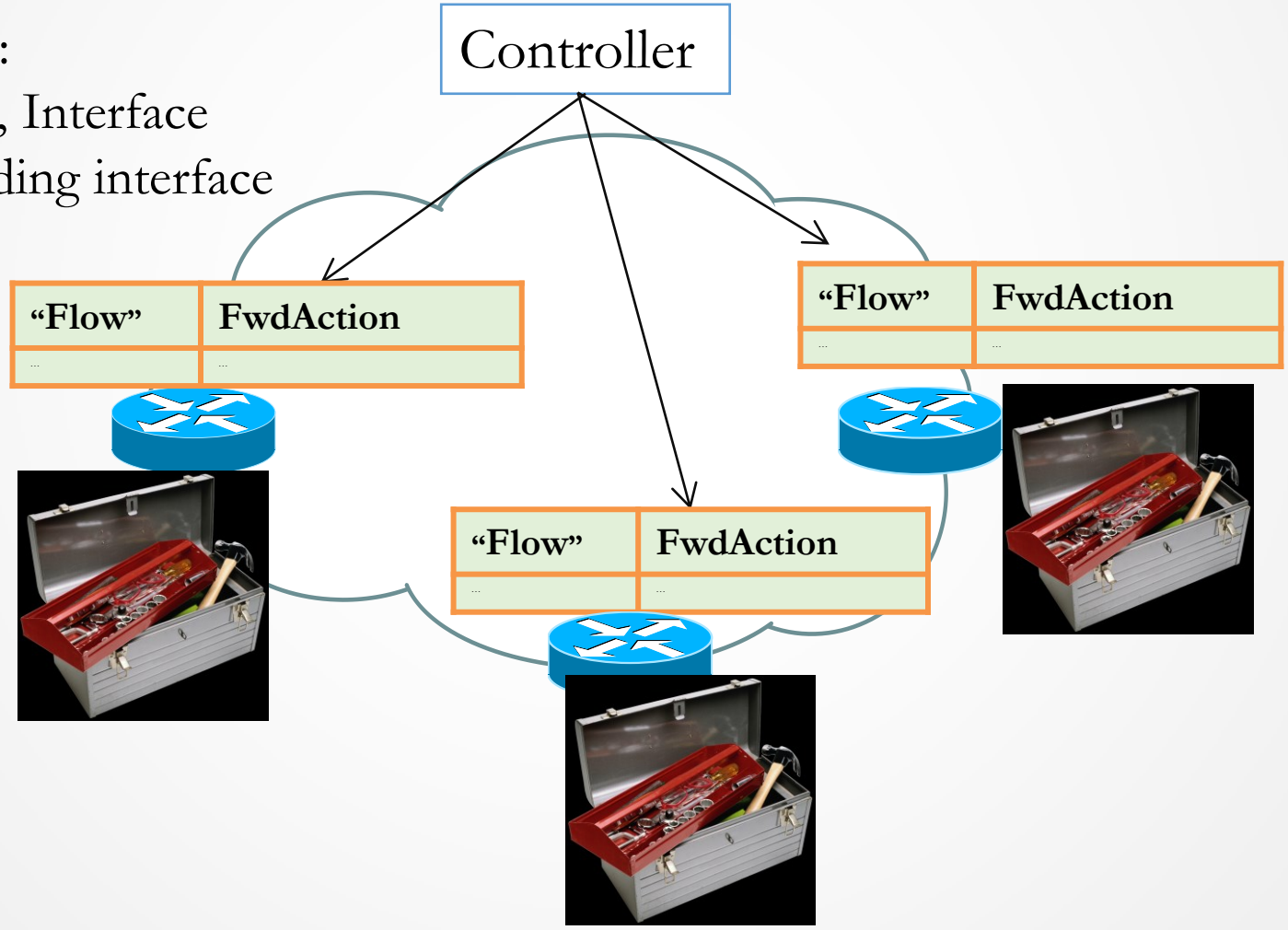
Vyas Sekar
Carnegie Mellon University

A 20000 feet view of Software-Defined Networking (SDN)

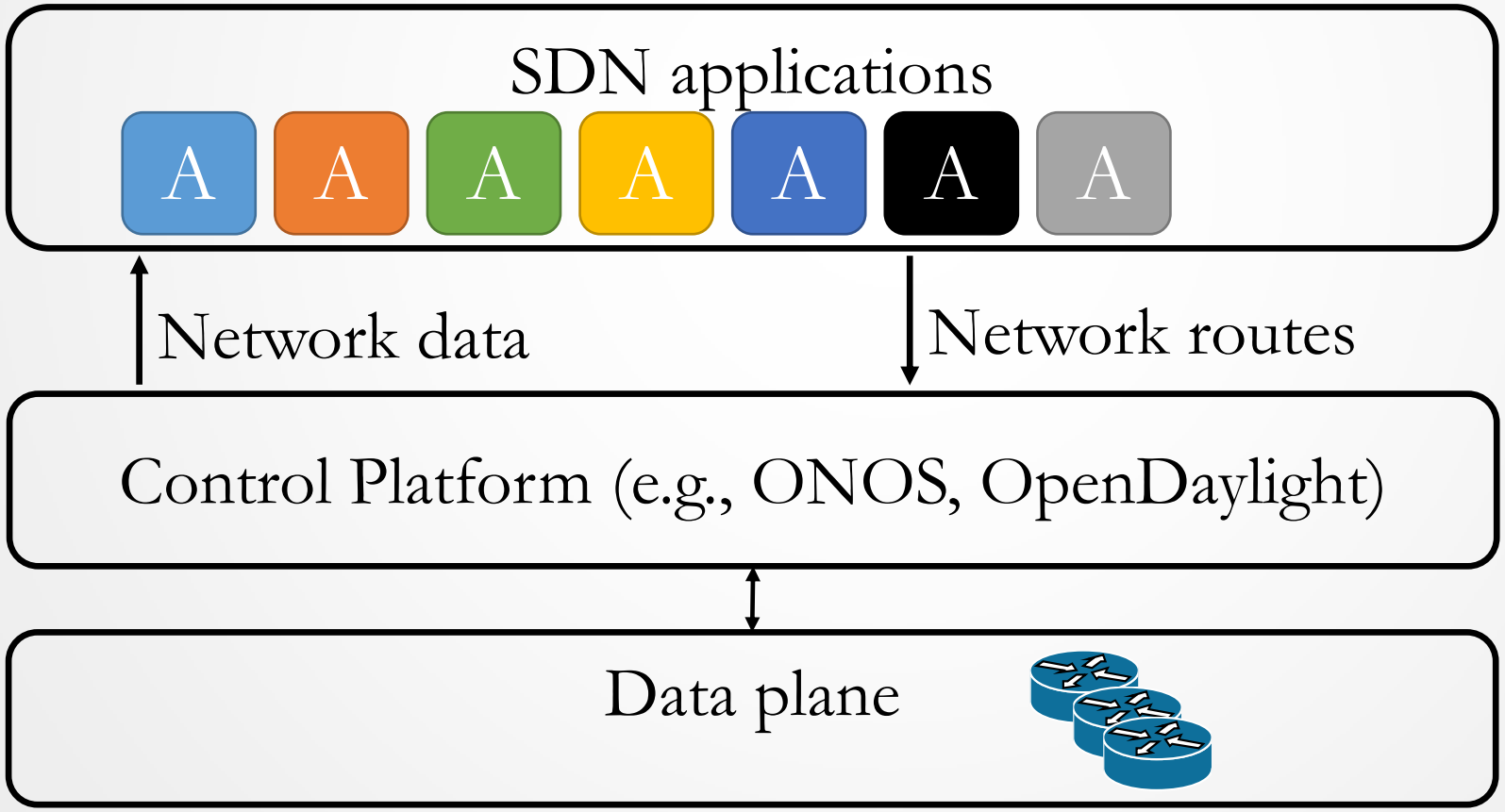
Centralized management + Open config APIs

OpenFlow:
Pkt header, Interface
→ Forwarding interface





What SDN looks like functionally?

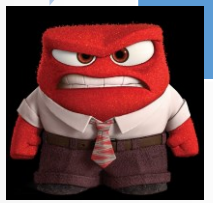


Network Optimizations are Common

- Maxflow, Traffic engineering
- SIMPLE (SIGCOMM 2013)
- ElasticTree (NSDI 2010)
- Panopticon (Usenix ATC 2014)
- SWAN (SIGCOMM 2013)

Current Process

Take theory & optimization courses

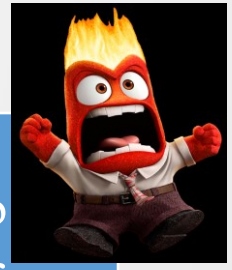


Formulate the problem

Solve with a solver

Not fast enough

- NP hard?

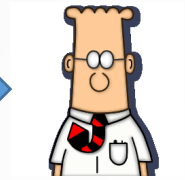


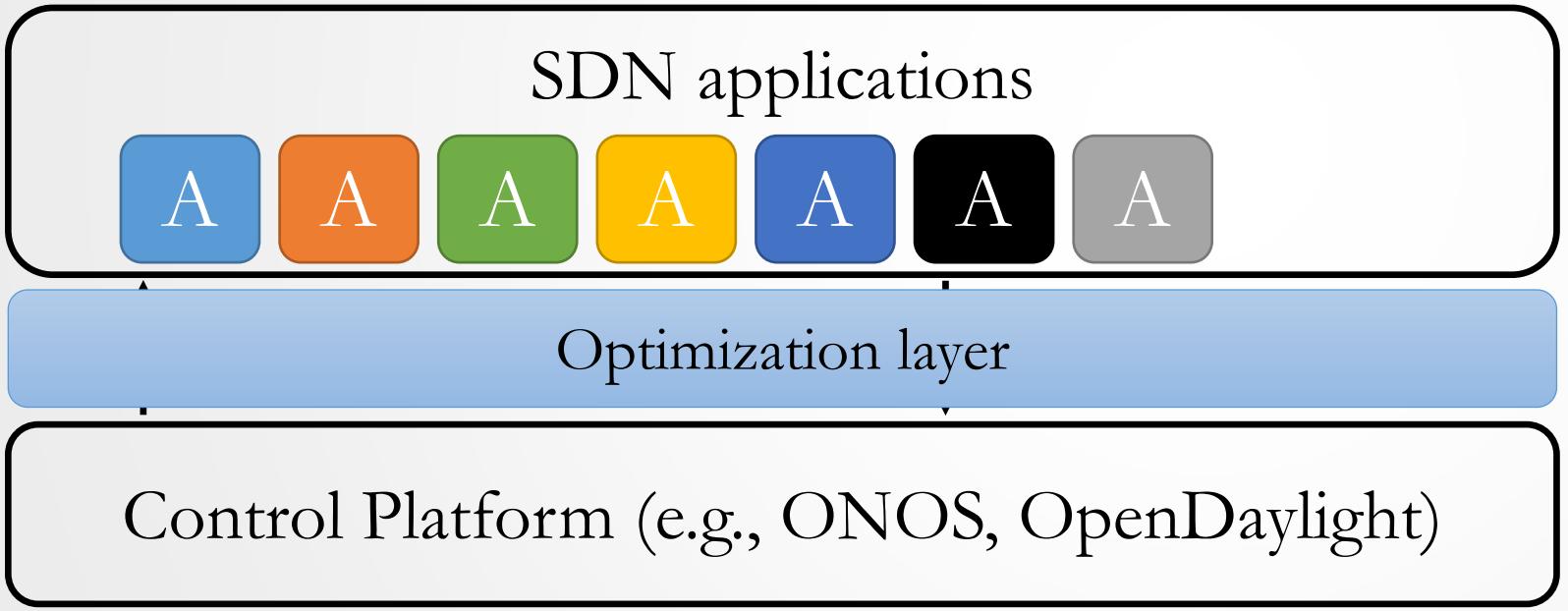
Develop heuristic

Parse solution

Deploy

Our Vision:
Practical Foundations for SDN optimization





- Focus on high-level network goals
- Rapid prototyping
- App = 20 lines of code

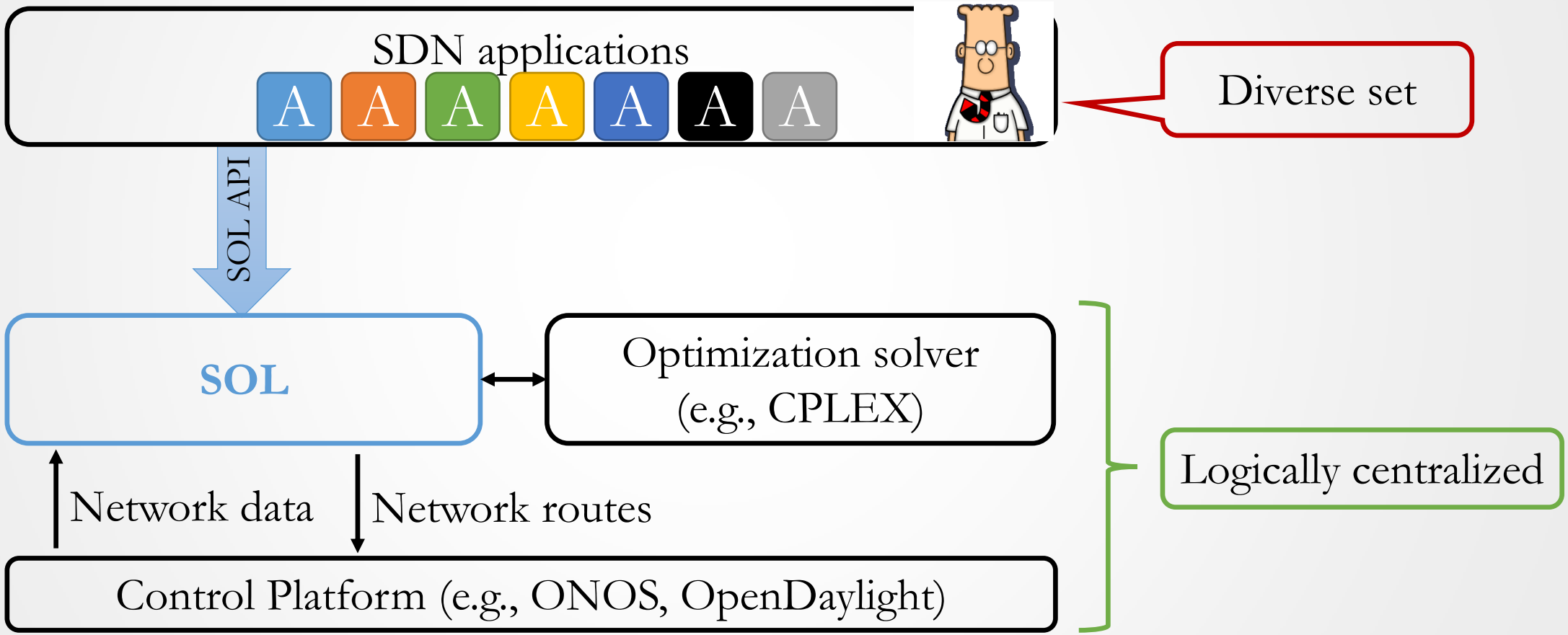
Project scope and goals

- Completed: Framework to simplify basic SDN app development
  - New abstractions and rule synthesis tools
  - [NSDI'16] paper and open source tool https://github.com/progwriter/SOL
- Future
  - Support composition of applications
  - Advanced abstractions beyond basic apps
  - Support for stochastic/adversarial demands

SOL Framework to simplify workflow

Approach	Generality	Efficiency
Frameworks	✓	✗
Custom solutions	✗	✓
SOL	✓	✓

SOL: **S**DN **O**ptimization **L**ayer



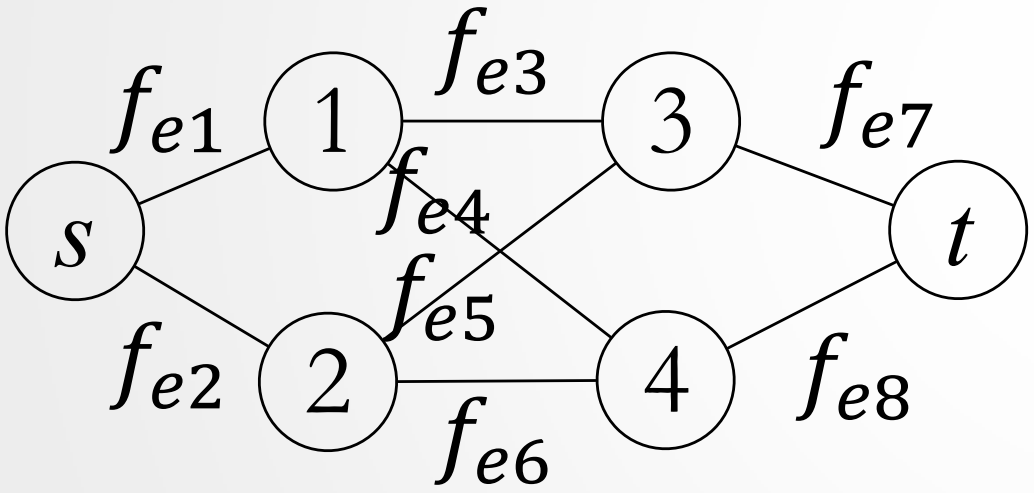
Insight: Path Abstraction

- Problems are *recast* to be **path-based**
- Policies are path predicates

Path-based Recasting: MaxFlow

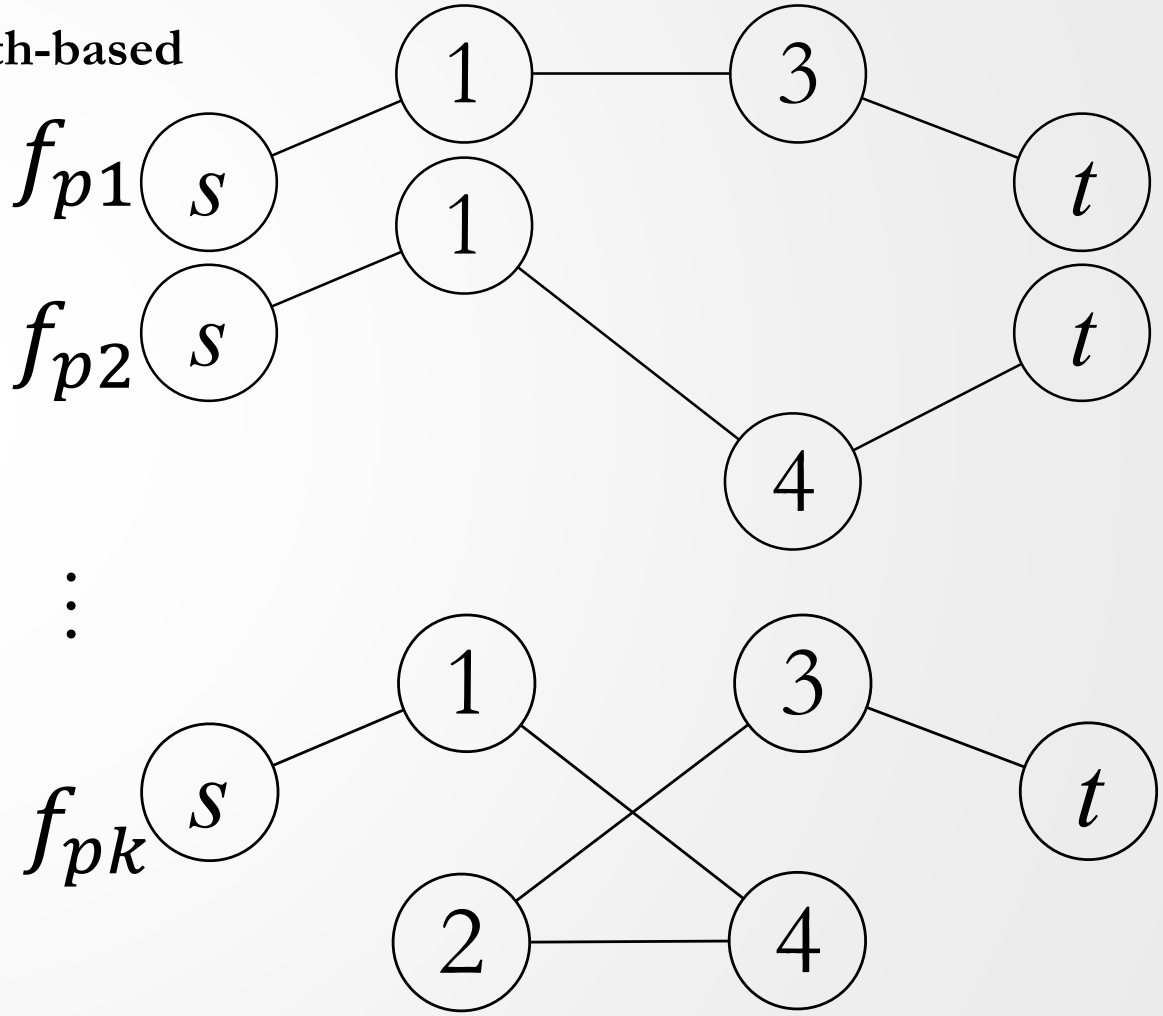
Edge-based

f: amount of flow



$$f_{e1} = f_{e3} + f_{e4}$$

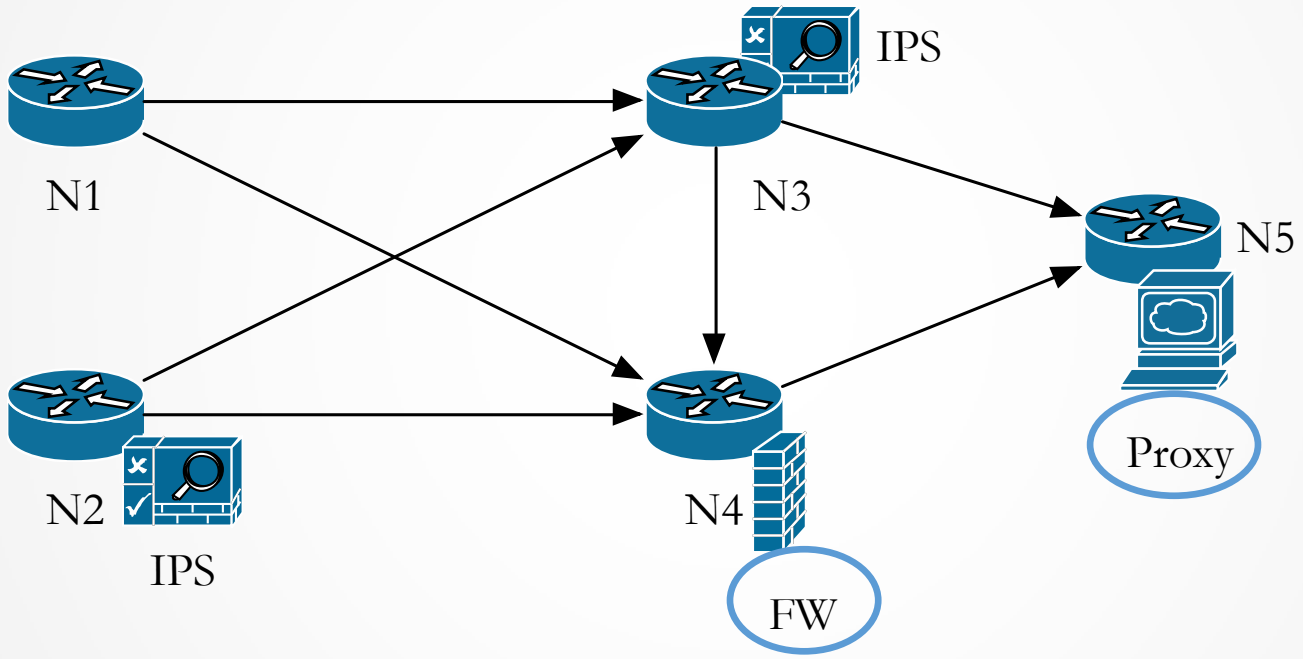
Path-based



$\sum_{i=1}^{k} f_{pi}$ = demand

Policies as Path Predicates

N1→N5
Web, 100 Mbps
FW→Proxy



Valid paths:

- N1-N4-N5
- N1-N3-N4-N5

Invalid paths:

- N1-N3-N5

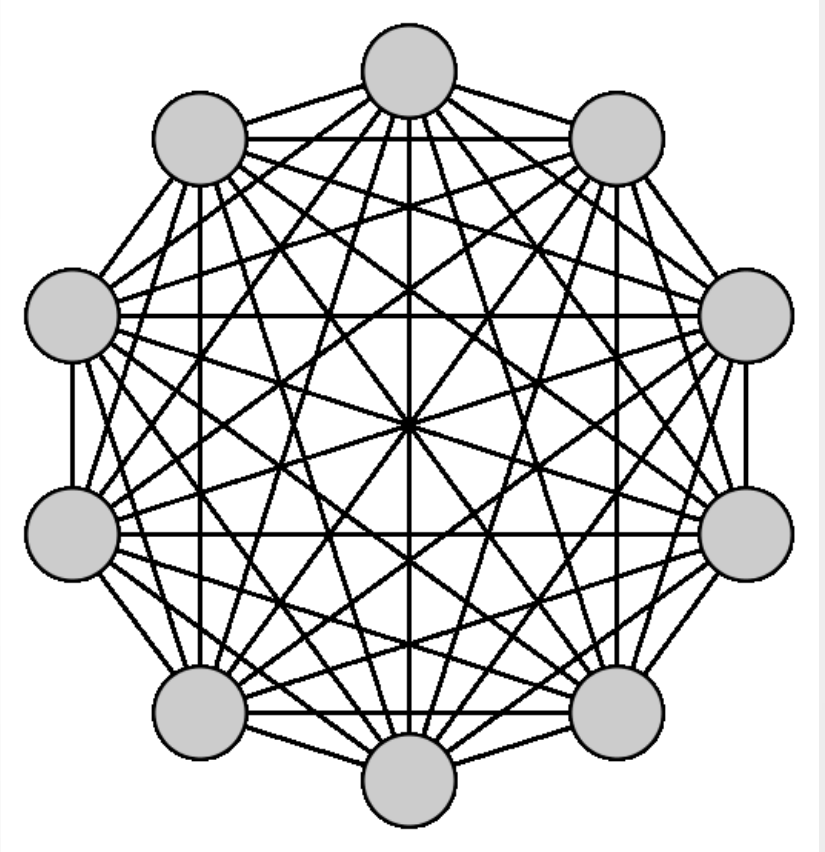
Generality

Path Challenge

Exponential number of paths

Large optimization size

Long run time = Bad efficiency



SOL Process

Path generation → Path selection → Optimization → Rule generation

1. Enumerate all simple paths
2. Keep valid paths (according to a predicate)

Offline step

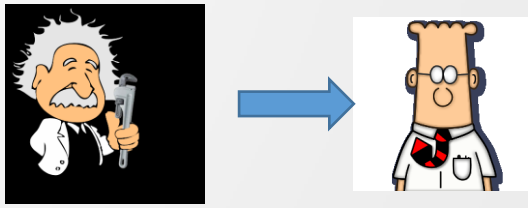
Pick a subset of paths

This acts as a **heuristic**

1. Model resource usage and constraints
2. Solve

Use a controller to configure data plane paths

Efficiency



Implementation

- Python library; interfaces with CPLEX solver and ONOS controller
- Prototyped applications
  - MaxFlow, Traffic engineering, latency minimization
  - ElasticTree (Heller et al.), Panopticon (Levin et al.), SIMPLE (Qazi et al.)

Example: MaxFlow

Topology input

Path generation + selection

```
1. opt, pptc = initOptimization(topo, trafficClasses, nullPredicate, 'shortest', 5)
2. opt.allocateFlow(pptc)
3. linkcapfunc = lambda link, tc, path, resource: tc.volBytes
4. opt.capLinks(pptc, 'bandwidth', linkConstrCaps, linkcapfunc)
5. opt.maxFlow(pptc)
6. opt.solve()
```

Traffic flows

Resource consumption

Global goal (objective function)

Example: Traffic Engineering

```
1. opt, pptc = initOptimization(topo, trafficClasses, nullPredicate, 'shortest', 5)
2. opt.allocateFlow(pptc)
3. linkcapfunc = lambda link, tc, path, resource: tc.volBytes
4. opt.capLinks(pptc, 'bandwidth', linkConstrCaps, linkcapfunc)
5. opt.routeAll(pptc)
6. opt.minLinkLoad('bandwidth')
7. opt.solve()
```

Route all traffic
Minimize bandwidth load

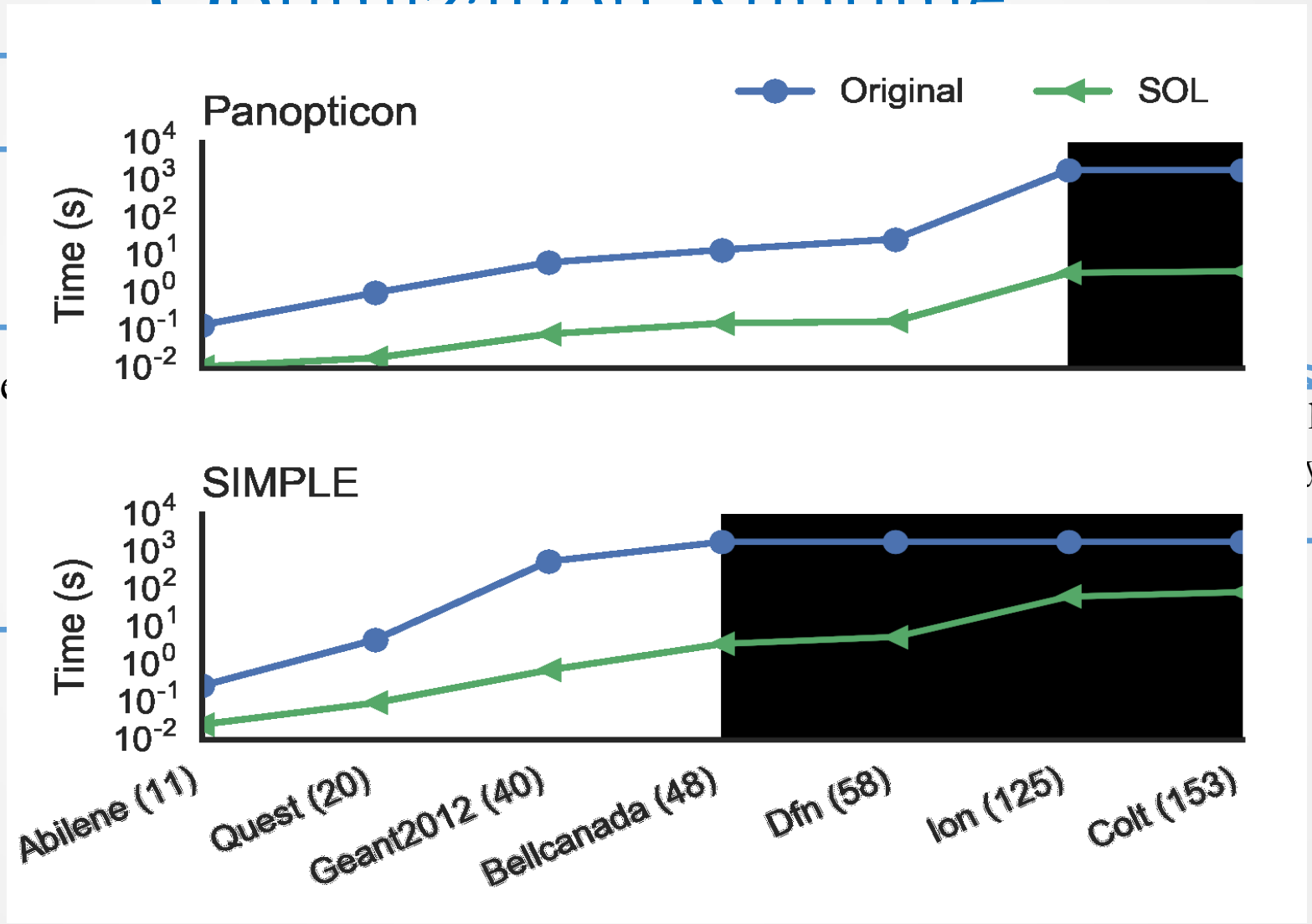
Development effort

Application	SOL lines of code	Estimated improvement
ElasticTree (Heller et al.)	16	21.8×
Panoption (Levin et al.)	13	25.7×
SIMPLE (Qazi et al.)	21	18.6×

Optimization Runtime

Log Scale

- Orders of magnitude **faster**
- Less than 1% away from **optimal**



haded: No solution
y the original within
30 minutes

Open questions and next steps?

- When/why does path pruning work?
- What is a good pruning strategy for a given objective/topology?
- Robustness to varying demands?
- Enabling composition of apps?
- Are paths sufficient or do we need richer abstractions?

Broader efforts in this space ..

AitF-funded workshop on Algorithms for Software-Defined Networking

Thanks to Mike Dinitz,Thyaga, Tracy, Rebecca Wright!

At DIMACS, Jun 2-3 2016

Program:
http://dimacs.rutgers.edu/Workshops/SDNAlgorithms/program.html

Videos!
https://www.youtube.com/playlist?list=PLqxsGMRlY6u7BhnI6JxShJHj_tYg-i1Qh

Conclusions

- SDN benefits in the field requires optimization
- Vision: Practical foundations for SDN optimization

  Lower barrier of entry for developers
- Initial work on SOL:
  - Leverages the path abstraction: generation + selection
  - Efficient: deploy in seconds!
- Enabler for new directions
  - E.g., seamless composition
- Many open theoretical questions with practical implications in SDN space