Cantor Meets Scott:

Semantic Foundations for Probabilistic Networks

Steffen Smolka (Cornell)
Praveen Kumar (Cornell)
Nate Foster (Cornell & Barefoot)
Dexter Kozen (Cornell)
Alexandra Silva (UCL)

's Wide Area Network

How to ensure correct behavior?

high level languages & automatic verification!

[Foster et al., ICFP 11], [Monsanto et al., POPL 12], [Kazemian et al., NSDI 12], [Voellmy et al., SIGCOMM 13], [Khurshid et al., NSDI 13], [Nelson et al., NSDI 14], [Anderson et al., POPL 14], [Plotkin et al., POPL 16], [Becket et al., PLDI 16], [Subramanian et al., POPL 17], …

Assumption: network behavior is **deterministic**

A language for **modeling & reasoning** about networks **probabilistically.**

ProbNetKAT

A language for **modeling & reasoning** about networks **probabilistically.**

Prob + NetKAT

probabilistic primitive

$p \oplus_r q$

network primitives

f:=n, dup

A language for **modeling & reasoning** about networks **probabilistically.**

ProbNetKAT 2016

NetKAT 2013

KAT 1996

1956 1847

Prob	+	Net	+	KA	+	T
probabilistic primitives		network primitives		regular expressions		boolean tests
$p \oplus_r q$		f:=n, dup		+, ·, *		f=n

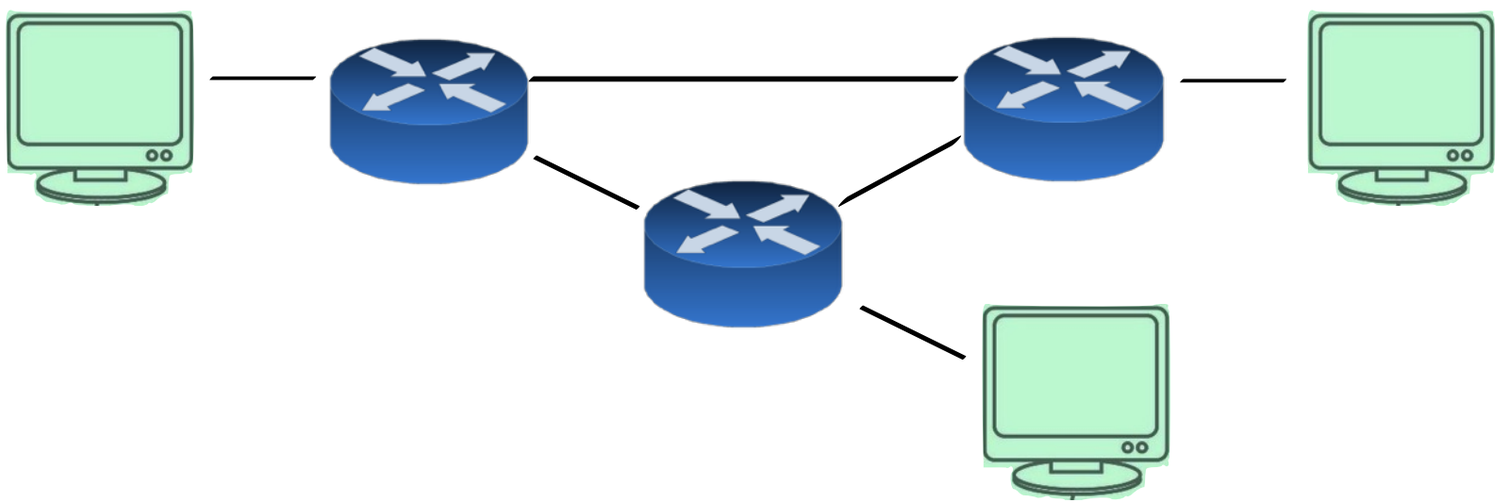
A language for **modeling & reasoning** about networks **probabilistically.**

$$[\![p]\!] \in 2^H \to \mathrm{Dist}(2^H)$$

$$[\![p]\!] \in 2^H \to 2^H$$

Prob	+	Net	+	KA	+	T
probabilistic primitives		network primitives		regular expressions		boolean tests
$p \oplus_r q$		f:=n, dup		+, ·, *		f=n

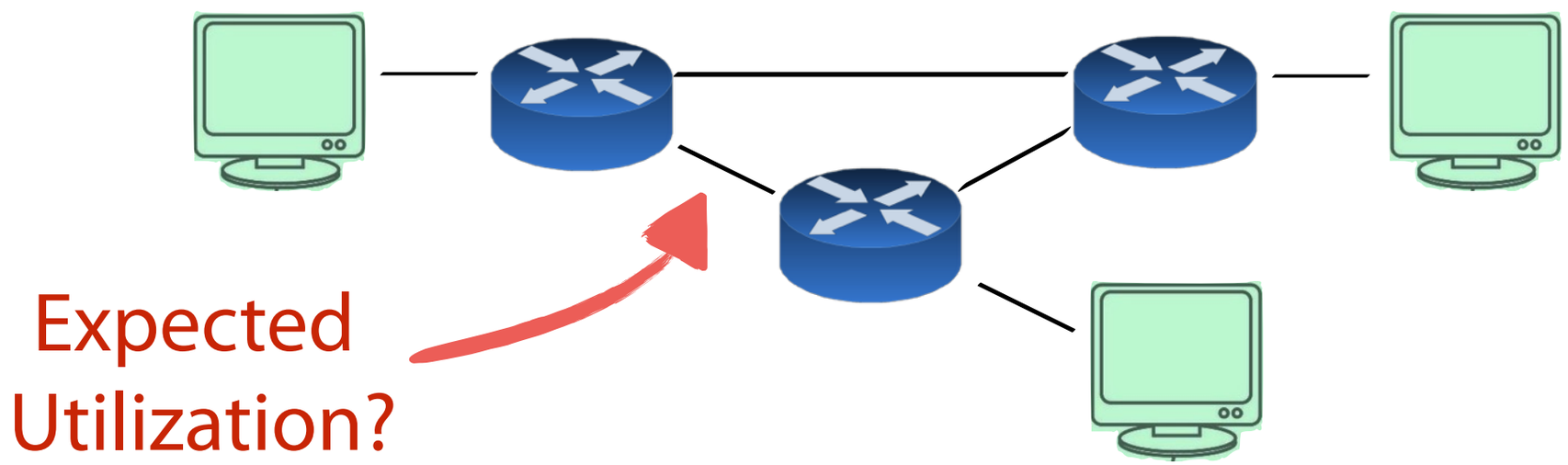
Probabilistic Reasoning



ProbNetKAT model **p**,
input distribution **μ**

→ traffic distribution **ν** = $[\![p]\!]^{\dagger}(\mu) \in \mathrm{Dist}(2^{H})$

Probabilistic Reasoning



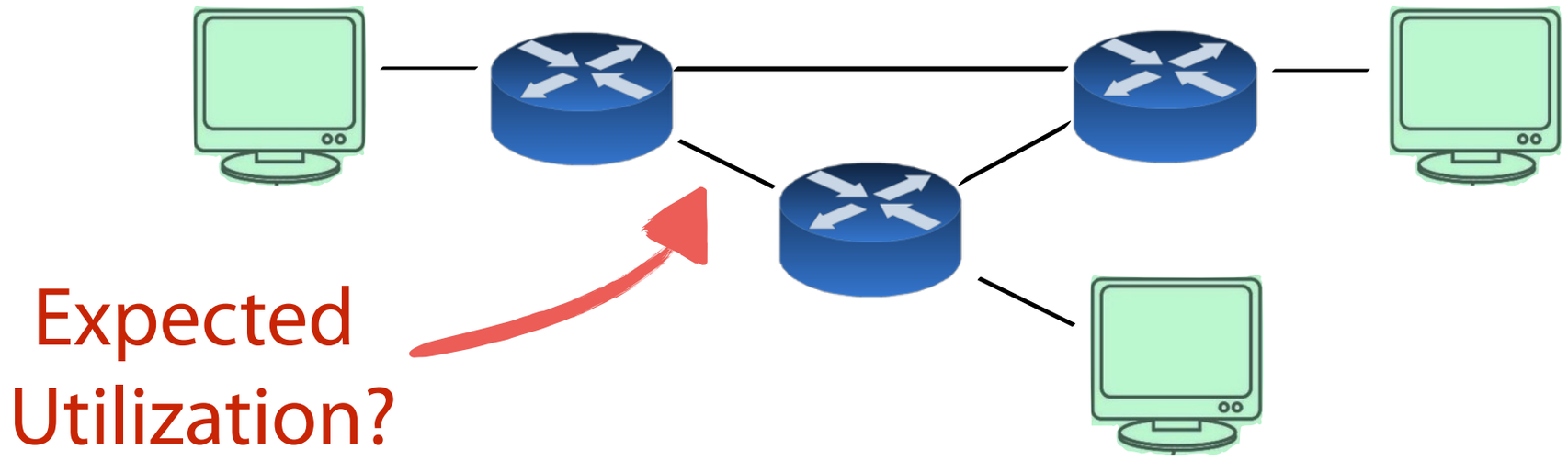
ProbNetKAT model **p**,
input distribution **μ**

→ traffic distribution **ν** = $[\![p]\!]^\dagger(\mu) \in \mathrm{Dist}(2^H)$

utilization query: **Q** : $2^H \to [0,\infty]$

expected utilization: $E_{\nu}[\mathbf{Q}]$

How to implement this?



$$E_{\nu}[\mathbf{Q}] = \int Q \, d\nu$$

Lebesgue Integral

continuous distribution over 2^H

Key Question: Approximation?

Key Idea

limits + continuity → approximation

$\mu_1, \mu_2, \mu_3, \ldots$ —**converges**→ $\mu \in \mathrm{Dist}(2^H)$

Key Idea

limits + continuity → approximation

$\mu_1, \mu_2, \mu_3, \ldots$ **converges** $\mu \in \mathrm{Dist}(2^H)$

$f(\mu_1), f(\mu_2), f(\mu_3), \ldots$ **converges** $f(\mu) \in R$

Key Idea

limits + **continuity** → approximation

$\mu_1, \mu_2, \mu_3, \ldots$ —**converges**→ $\mu \in \mathrm{Dist}(2^H)$

continuity (↓)

$f(\mu_1), f(\mu_2), f(\mu_3), \ldots$ —**converges**→ $f(\mu) \in R$

Main Results

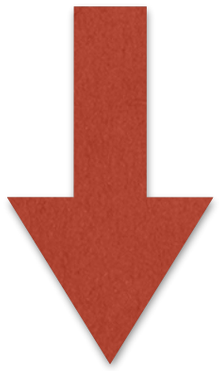
1) Iteration-free programs generate only finite distributions
2) Iteration may introduce continuous distributions …
   … but can be approximated by bounded iteration

3) All programs can be approximated

Main Results

1) Iteration-free programs generate only finite distributions
2) Iteration may introduce continuous distributions …
   … but can be approximated by bounded iteration

compositionality of approximation



continuity of $+, \cdot, ^*, \oplus$

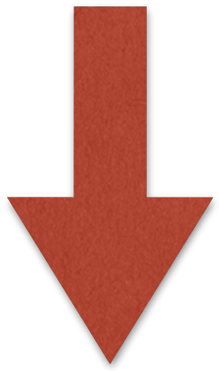
3) All programs can be approximated

Main Results

1) Iteration-free programs generate only finite distributions

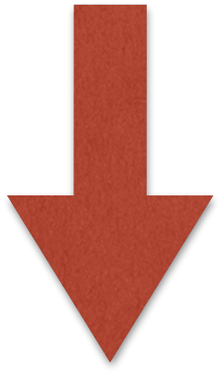
2) Iteration may introduce continuous distributions …
… but can be approximated by bounded iteration

compositionality of approximation



continuity of $+, \cdot, ^*, \oplus$

3) All programs can be approximated



continuity of $\int, \mathbb{E}[-]$

4) Queries can be approximated

Main Results

1) Star-free programs generate only finite distributions

2) Iteration may introduce continuous distributions …
… but can be approximated by bounded iteration

but different topologies give different notions of limits, continuity, and approximation

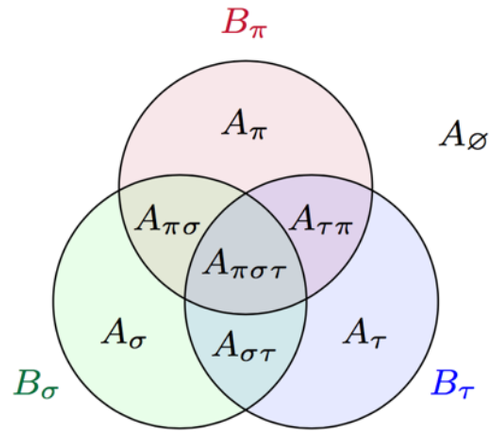
3) All programs can be approximated

continuity of
$\int$, E[-]

4) Queries can be approximated

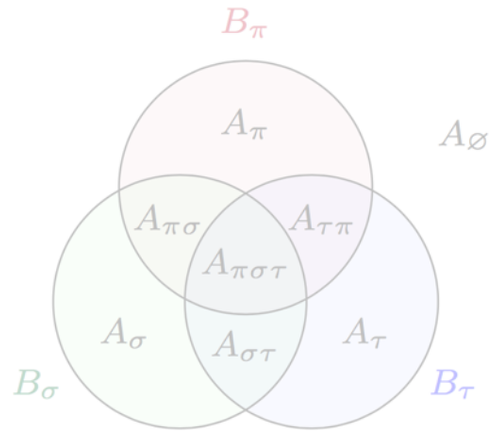
Cantor Meets Scott

Topologies



	Cantor	Scott
	←d(μ,ν)→	μ ⊑ ν
Metric	✔	✘
Convergence	**Weak**	Monotone
Practical Queries	✘	✔

Topologies



	Cantor	Scott
	$Q : 2^H \rightarrow R$ $Q(A) = \lvert A \rvert$ "network congestion"	
Convergence	Weak	Monotone
Practical Queries	✗	✔

CPOs for ProbNetKAT

If a larger set of packets (in the sense of ⊆) is input to the ProbNetKAT program, then the probability that a given set of packets occurs as a subset of the output set can only increase.

history sets	*distributions*	*programs*	*query results*
$(2^H, \subseteq)$	$(D(2^H), \sqsubseteq)$ [Saheb-Djahromi, Jones & Plotkin]	$(2^H \to D(2^H), \sqsubseteq)$	$(R, \leq)$

+, ·, *, ⊕, E-[-] respect this order!

Summary

→ any **program** is approximated to arbitrary precision by **finite distributions**!

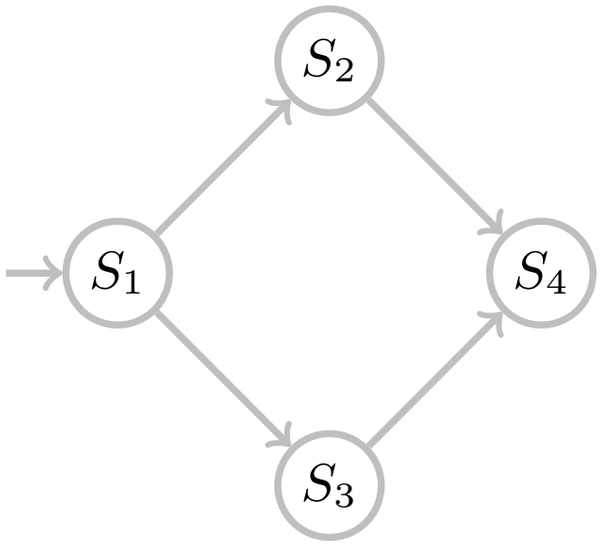
→ any **query** is approximated to arbitrary precision by **finite sums**!

→ convergence is **monotone**!

→ implemented in OCaml in ~300 LOC

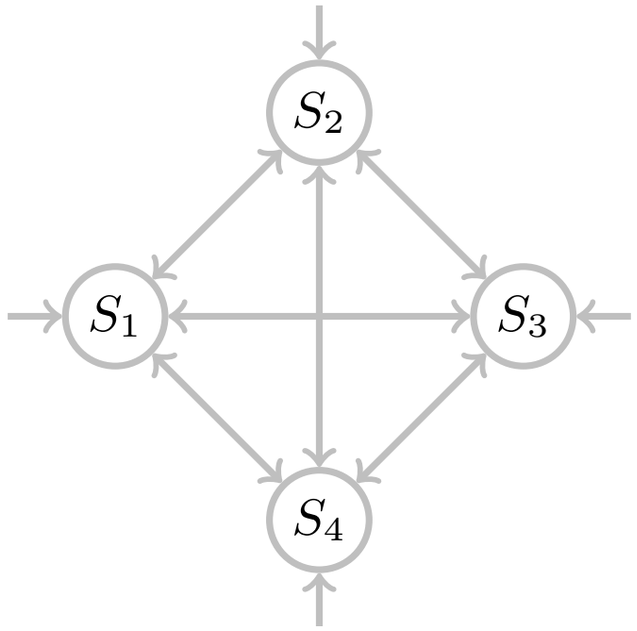
Applications

Fault Tolerance



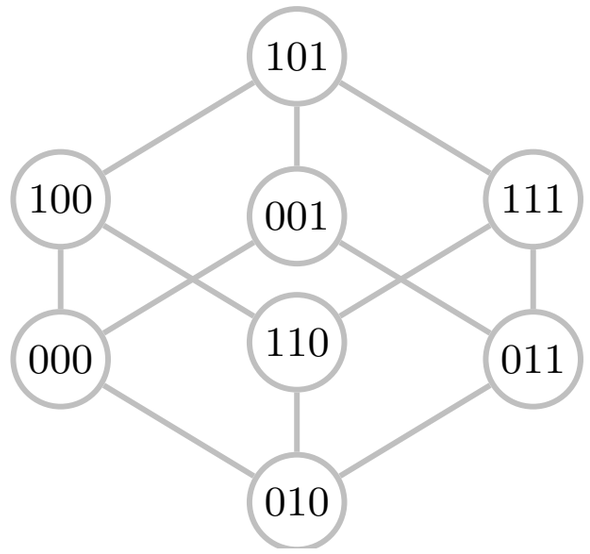
Probability of delivery in the presence of failures

Utilization

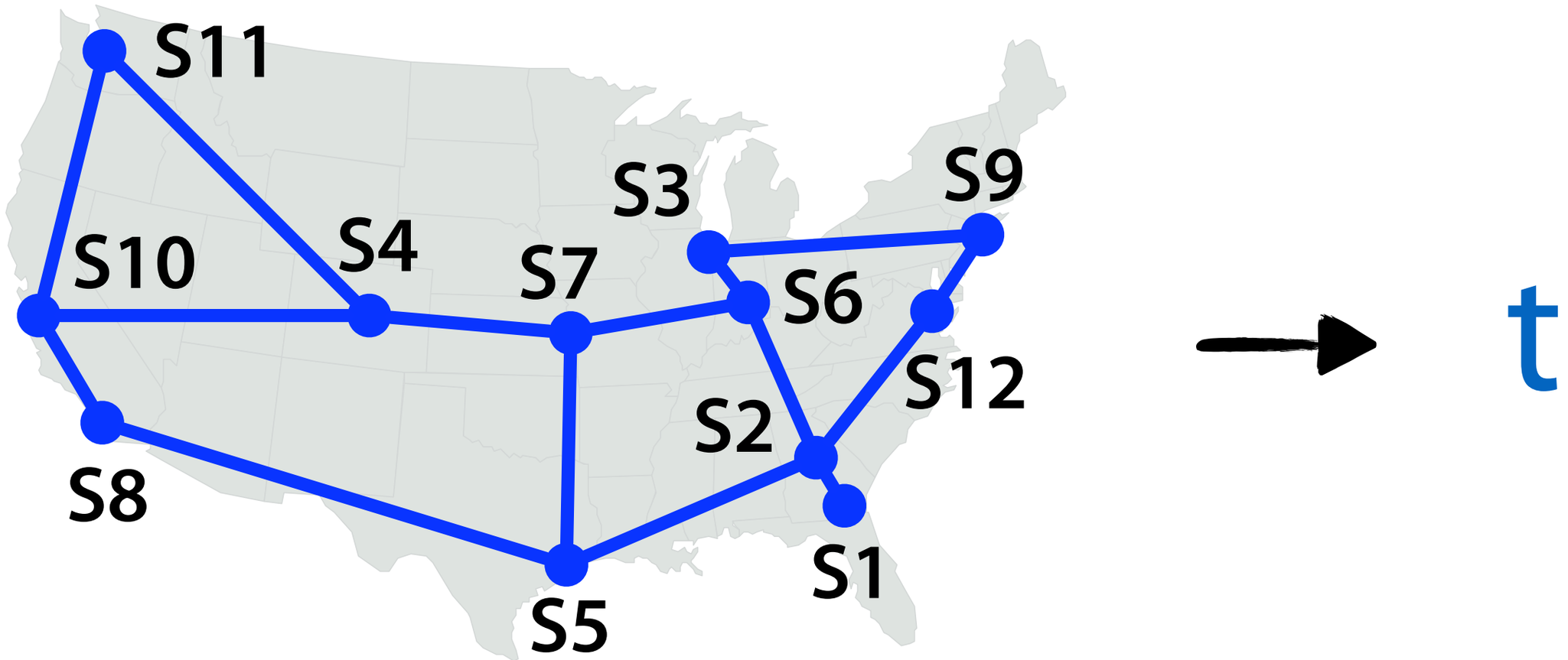


Expected number of packets traversing each link

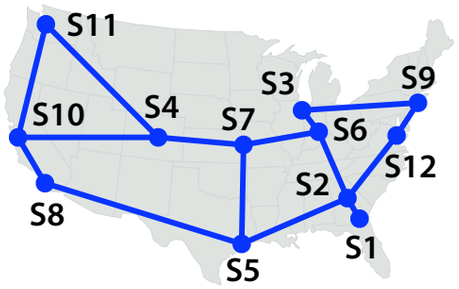
Gossip protocols



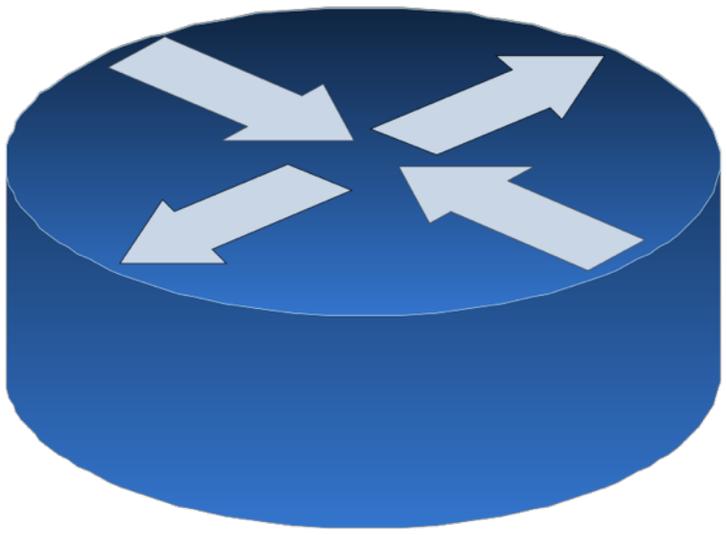
Expected number of nodes "infected" after n rounds



Topology



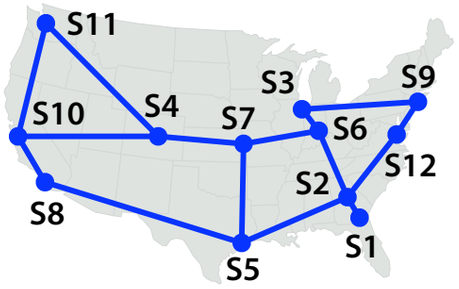
t



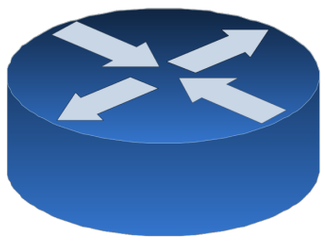
p

ECMP, KSP, Multi, Räcke

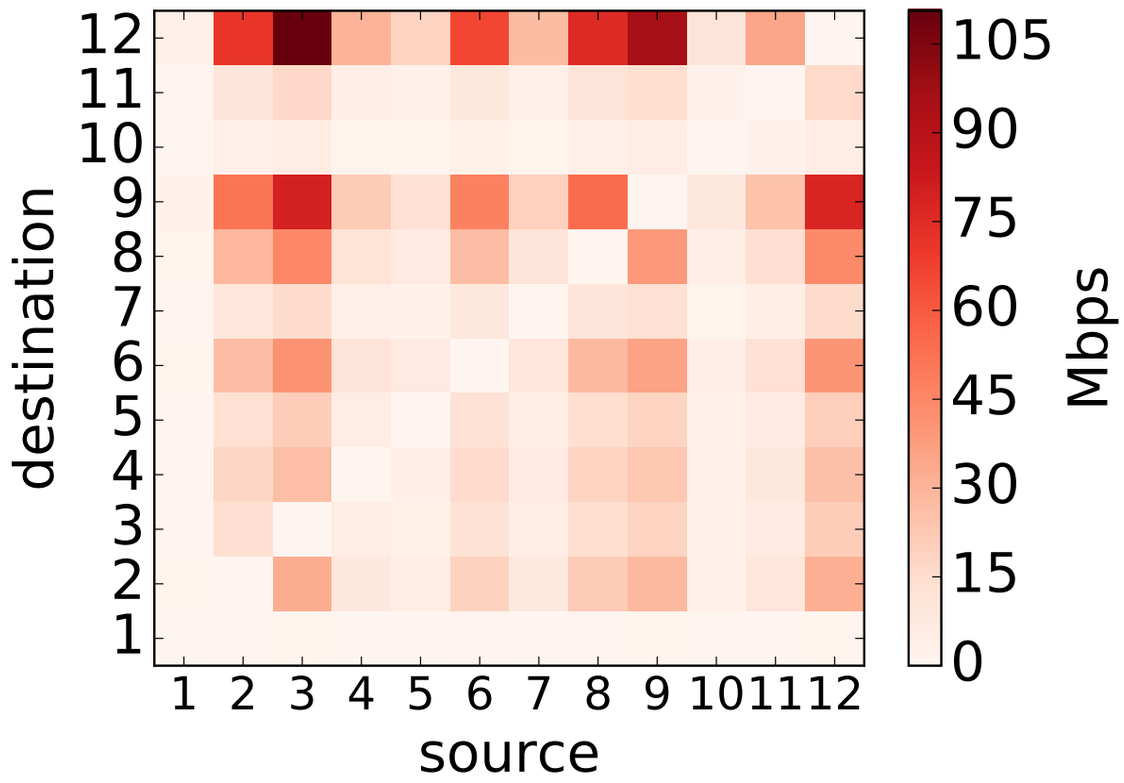
Routing Algorithms

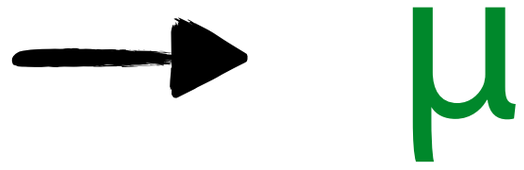


t



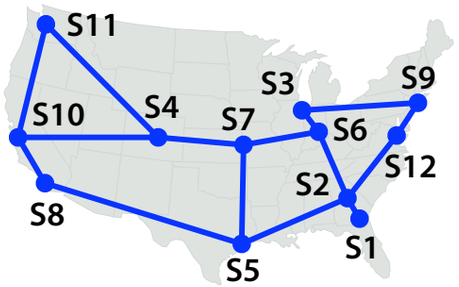
p



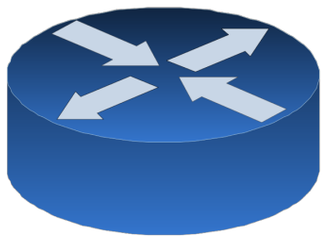


μ

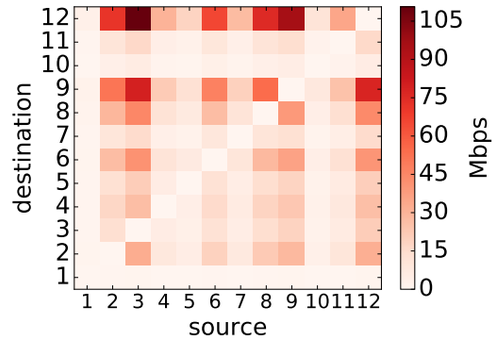
Demand Matrix



t



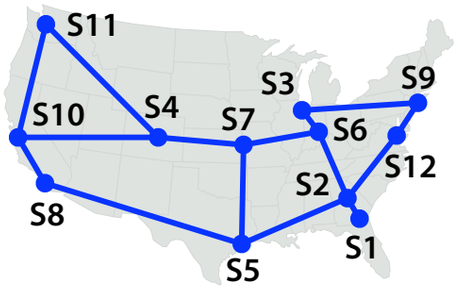
p



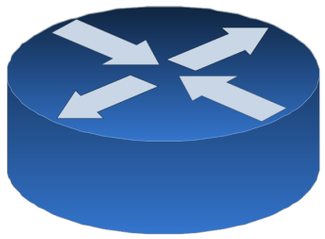
μ

$$(p \cdot t)^* \cdot p$$

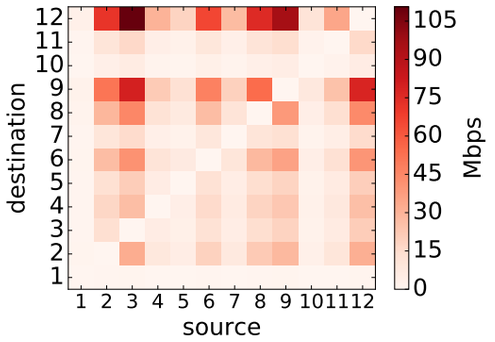
1. Assemble ProbNetKAT Model



t



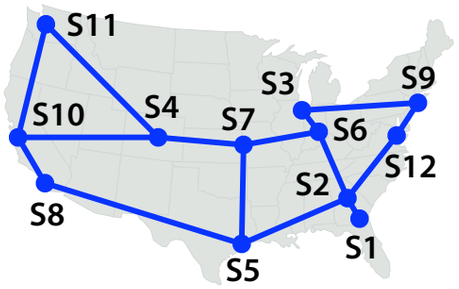
p



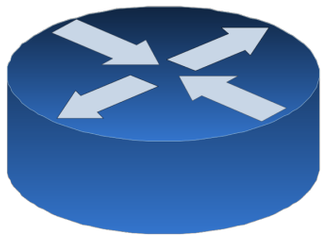
μ

$$[\![(p \cdot t)^* \cdot p]\!]_1(\mu) = \nu_1$$

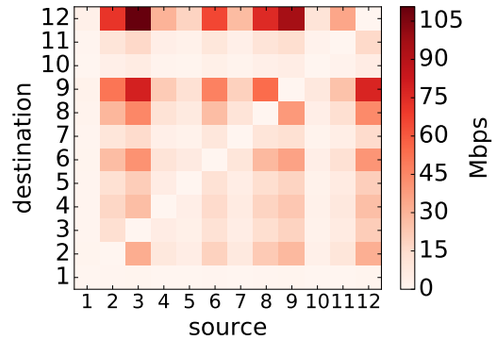
2. Approximate Traffic Distribution



t



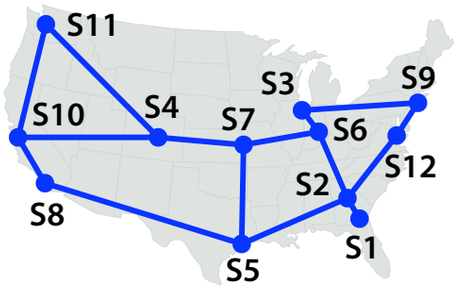
p



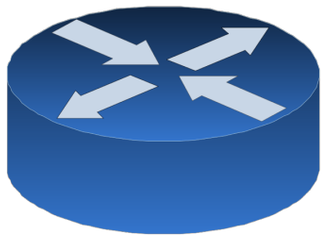
μ

$$[\![(p \cdot t)^* \cdot p]\!]_2(\mu) = \nu_2$$

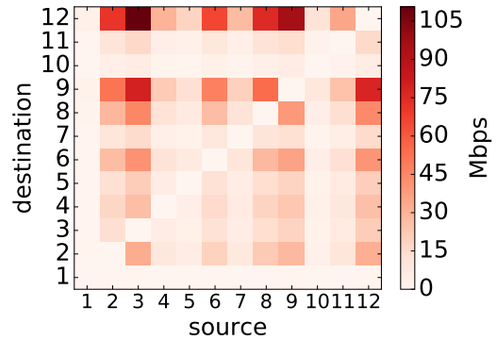
2. Approximate Traffic Distribution



t



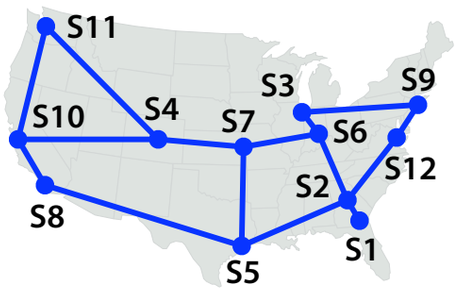
p



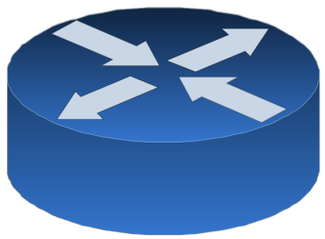
μ

$$[\![(p \cdot t)^* \cdot p]\!]_3(\mu) = \nu_3$$

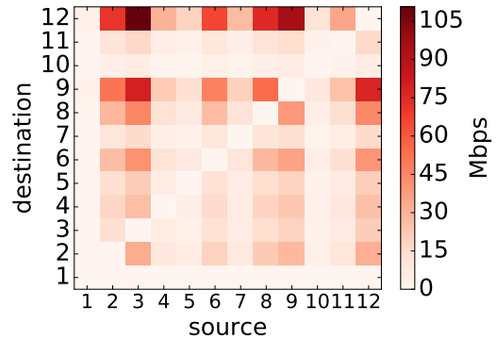
2. Approximate Traffic Distribution



t



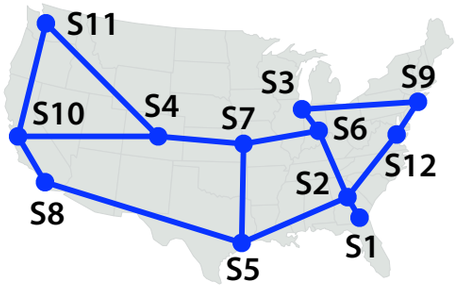
p



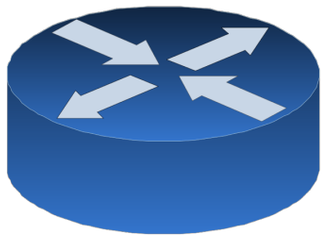
μ

$$\nu_1 \sqsubseteq \nu_2 \sqsubseteq \nu_3 \sqsubseteq \nu_4 \sqsubseteq \ldots$$

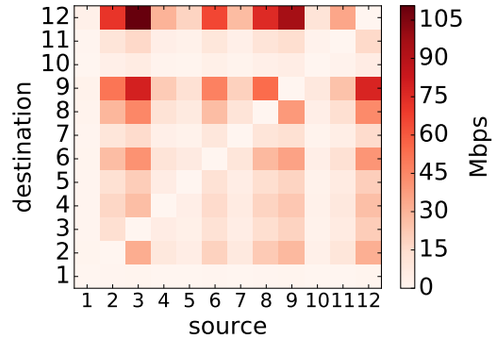
2. Approximate Traffic Distribution



t



p

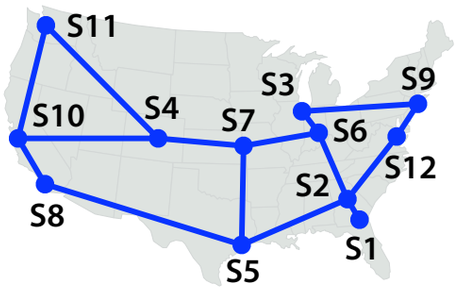


μ

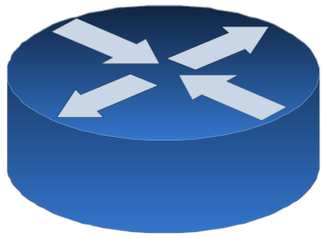
$E_{v1}[\text{hopcount}] \leq E_{v2}[\text{hopcount}] \leq \ldots$

```
1 hopcount : history -> int
2 hopcount h = List.length h
```

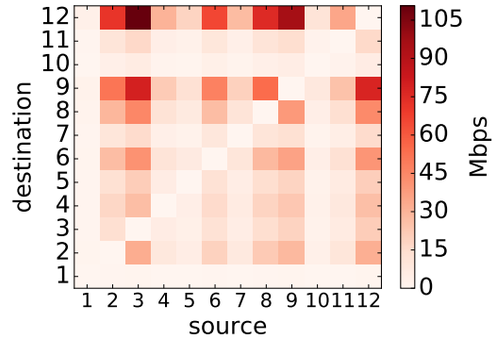
3. Approximate Network Metrics


S11
S3
S9
S10
S4
S7
S6
S12
S2
S8
S1
S5

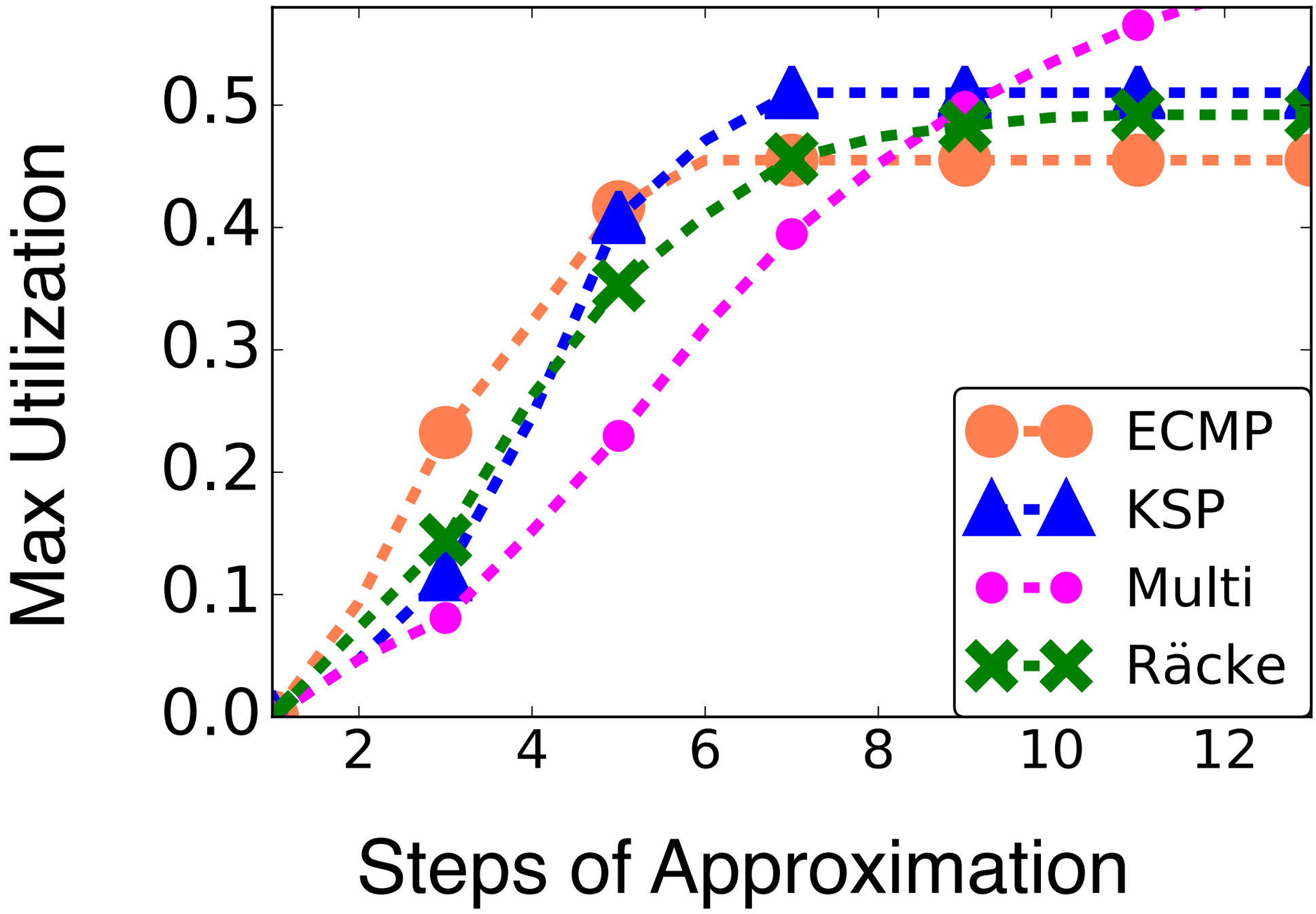
t



p


destination
source
Mbps
1 2 3 4 5 6 7 8 9 10 11 12
0 15 30 45 60 75 90 105

μ

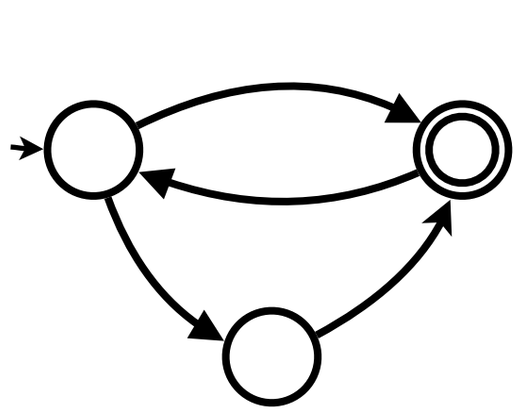

Max Utilization
0.0
0.1
0.2
0.3
0.4
0.5
2
4
6
8
10
12
Steps of Approximation
ECMP
KSP
Multi
Räcke

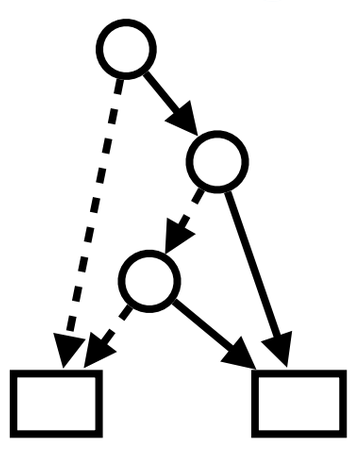
Ongoing Work

Richer language (e.g. link capacities, queuing, etc.)

$$A \rightarrow B;\ @1\text{Gbit/s}$$

Efficient implementation that scales to large networks





Axiomatic reasoning and a decision procedure

$$\vdash p \equiv q$$

Steffen
Smolka

Praveen
Kumar

Nate
Foster

Dexter
Kozen

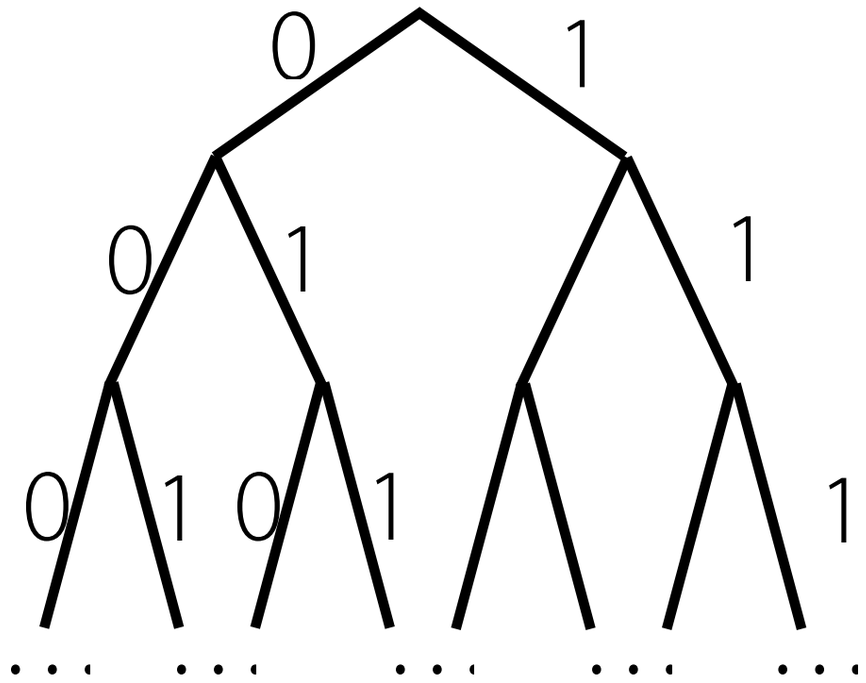
Alexandra
Silva

PROBABILISTIC

NETKAT

A continuous distribution

$$((\pi_0! \oplus \pi_1!) \bullet \mathbf{dup})^*$$



execution

≅

infinite path

≅

random output

$\in 2^H$

How many paths are there? → one for every $r \in [0,1]$

What's the probability of any particular path? → 0

Taming *

Recall: $[\![p]\!] \in 2^H \to \mathrm{Dist}(2^H)$

What is $[\![p^*]\!]$?

$$[\![p^*]\!](a) := \mu_Y$$

$$X_0 := a$$

$$X_{n+1} \sim [\![p]\!](X_n)$$

$$Y := X_0 \cup X_1 \cup X_2 \cup \ldots$$

Idea: stop executing loop after n iterations

- $Y_n := X_0 \cup \ldots \cup X_n$
- $[\![p^*]\!](a) := \text{“}\lim_n \mu_{Y_n}\text{”}$