Adriaan van Wijngaarden meets Scott
Domain-Theoretic Foundations for Probabilistic Network Programming

Alexandra Silva (UCL)
The gang
Networks

Network configuration largely a black art

- built and programmed the same way since the 1970s
- Difficult to implement end-to-end routing policies and optimisations that require a global perspective
- Difficult to extend with new functionality
- Effectively impossible to reason precisely about behaviour

configured locally using proprietary interfaces
Software-Defined Networks

Your Program goes here!

Ox Controller Platform
or POX, Beacon, Floodlight, others

OpenFlow API

OpenFlow-compatible switches
Pica8, Dell, NEC, HP, many others
Openflow

- Specifies capabilities and behaviour of switch hardware
- A language for manipulating network configurations
- Very low-level: easy for hardware to implement, difficult for humans to write and reason about

But…

✓ is platform independent
✓ provides an open standard that any vendor can implement

[McKeown & al., SIGCOMM 08]
Verification of networks

Trend in PL Verification after Software-Defined Networks

• Design *high-level languages* that model essential network features
• Develop *semantics* that enables reasoning precisely about behaviour
• Build *tools* to synthesise low-level implementations automatically

- Frenetic [Foster & al., ICFP 11]
- Pyretic [Monsanto & al., NSDI 13]
- Maple [Voellmy & al., SIGCOMM 13]
- FlowLog [Nelson & al., NSDI 14]
- Header Space Analysis [Kazemian & al., NSDI 12]
- VeriFlow [Khurshid & al., NSDI 13]
- NetKAT [Anderson & al., POPL 14]
- and many others . . .
NetKAT

NetKAT

=  

Kleene algebra with tests (KAT)

+  

additional specialized constructs particular to network topology and packet switching

Kleene Algebra (KA)

\[(0 + 1(01^*0)^*1)^*\]  
\[\{\text{multiples of 3 in binary}\}\]

\[(ab)^* a = a(ba)^*\]  
\[\{a, aba, ababa, \ldots\}\]

\[(a + b)^* = a^*(ba^*)^*\]  
\[\{\text{all strings over } \{a, b\}\}\]
(K, B, +, ·, *, −, 0, 1), B ⊆ K

- (K, +, ·, *, 0, 1) is a Kleene algebra
- (B, +, ·, −, 0, 1) is a Boolean algebra
- (B, +, ·, 0, 1) is a subalgebra of (K, +, ·, 0, 1)

KAT = simple imperative language

- If b then p else q = b;p + !b;q
- While b do p = (bp)*!b

NetKAT
a packet $\pi$ is an assignment of constant values $n$ to fields $x$

a packet history is a nonempty sequence of packets

$\pi_1 :: \pi_2 :: \cdots :: \pi_k$

the head packet is $\pi_1$

assignments $x \leftarrow n$

assign constant value $n$ to field $x$ in the head packet

tests $x = n$

if value of field $x$ in the head packet is $n$, then pass, else drop

dup

duplicate the head packet
Networks in NetKAT

\[ \text{sw}=6; \text{pt}=8; \text{dst} := 10.0.1.5; \text{pt}=5 \]

For all packets located at port 8 of switch 6, set the destination address to 10.0.1.5 and forward it out on port 5.
Networks in NetKAT

The behaviour of an entire network can be encoded in NetKAT by interleaving steps of processions by switches and topology.
Semantics

packet history $\langle p, \ldots \rangle$  

$(\text{policy};\text{topo})^*;\text{policy}$  

set of packet histories $\{\langle q, \ldots \rangle, \langle r, \ldots \rangle\}$

\[ [e] : H \rightarrow 2^H \]

\[ [x \leftarrow n](\pi_1 :: \sigma) \triangleq \{\pi_1[n/x] :: \sigma\} \]

\[ [x = n](\pi_1 :: \sigma) \triangleq \begin{cases} \{\pi_1 :: \sigma\} & \text{if } \pi_1(x) = n \\ \emptyset & \text{if } \pi_1(x) \neq n \end{cases} \]

\[ [\text{dup}](\pi_1 :: \sigma) \triangleq \{\pi_1 :: \pi_1 :: \sigma\} \]
Verification using NetKAT

Reachability
- Can host A communicate with host B? Can every host communicate with every other host?

Security
- Does all untrusted traffic pass through the intrusion detection system located at C?

Loop detection
- Is it possible for a packet to be forwarded around a cycle in the network?
Verification using NetKAT

Soundness and Completeness [Anderson et al. 14]

- ⊢ p = q if and only if \([p] = [q]\)

Decision Procedure [Foster et al. 15]

- NetKAT coalgebra
- efficient bisimulation-based decision procedure
- implementation in OCaml
- deployed in the Frenetic suite of network management tools
Limitations

\[ [e] : H \rightarrow 2^H \]

- Packet-processing function
- Applicability limited to simple connectivity or routing behavior

Probabilities are needed:
- expected congestion
- reliability
- randomized routing
ProbNetKAT

\[ p \ominus_r q \]

dst = h_3; pt \leftarrow 2 \ominus 0.5 pt \leftarrow 4
ProbNetKAT by example

\[ \text{net} \triangleq \text{in} ; (p ; t)^* ; p ; \text{out} \]

Ingress: egress, pt \leftarrow 1
\& (\text{dst} = h_2, \text{pt} = 2)
\& (\text{dst} = h_3, \text{pt} = 2)
\& (\text{dst} = h_4, \text{pt} = 4)
\& \ldots

Forwarding policy
\[ p \triangleq (\text{sw} = S_1 ; p_1) \& (\text{sw} = S_2 ; p_2) \& (\text{sw} = S_3 ; p_3) \& (\text{sw} = S_4 ; p_4) \]

Topology
\[ t \triangleq l_{1,2} \& l_{2,3} \& l_{3,4} \& l_{1,4} \]
Semantics

\[ [p] \in 2^H \rightarrow \{ \mu : B \rightarrow [0, 1] \mid \mu \text{ is a probability measure} \} \]

\[ [x \leftarrow n](a) = \delta_{\{ \pi[n/x]: \pi : \sigma \in a \}} \]
\[ [x = n](a) = \delta_{\{ \pi : \sigma \mid \pi : \sigma \in a, \pi(x)=n \}} \]
\[ [\text{dup}](a) = \delta_{\{ \pi : \pi : \sigma \mid \pi : \sigma \in a \}} \]
\[ [\text{skip}](a) = \delta_a \]
\[ [\text{drop}](a) = \delta_{\emptyset} \]

\[ [p \& q](a) = [p](a) \& [q](a) \]
\[ (\mu \& \nu)(A) \triangleq (\mu \times \nu)(\{(a, b) \mid a \cup b \in A\}). \]

\[ [p +_r q](a) = r[p](a) + (1 - r)[p](a) \]
Semantics

\[ [p] \in 2^H \rightarrow \{ \mu : B \rightarrow [0,1] \mid \mu \text{ is a probability measure} \} \]

\[ \begin{bmatrix} p^* \end{bmatrix} = \diamond \]

Ideally: \[ [p^*] = [1 \& pp^*] \]

least fix point? which order?

Ad-hoc attempt: infinite stochastic process
Semantics

ProbNetKAT model $p$, input distribution $\mu$

$\rightarrow$ output distribution $v = \mu \gg [p] \in \text{Dist}(2^H)$

Congestion Query: Random Variable $Q : 2^H \rightarrow [0,\infty]$

$$Q(a) \triangleq \sum_{h \in a} \#_i(h)$$

Expected Congestion: $E_v[Q]$

$$E_v[Q] = \int Q \, dv$$
Issues with previous semantics

No practical implementation?

Lebesgue Integral

Challenges in representing infinite distributions

Iteration — infinite stochastic process instead of standard fixpoint

weak convergence non-monotonic

continuous distribution

many queries not continuous Cantor topology — no weak convergence!
The importance of continuity

\[ a_1, a_2, \cdots \quad \xrightarrow{\text{limit}} \quad a \]

simple (e.g. finite) objects

\( a \) can be approximated by \( (a_n) \)

Perform computation \( f \) on \( a \)

\( f \) continuous

\[ f(a_1), f(a_2), \cdots \quad \xrightarrow{\text{limit}} \quad f(a) \]
The importance of continuity for network analysis

\[ \mu_1, \mu_2, \ldots \rightarrow \mu \]

finite support!

\[ E_\mu[f] \quad — \text{expected value of a continuous map is continuous} \]

monotonically improving sequence of approximations for performance metrics such as latency and congestion
New semantics

\([p] \in 2^H \rightarrow \{\mu : \mathcal{B} \rightarrow [0, 1] \mid \mu \text{ is a probability measure}\}\)

\([p^*] = \operatorname{lfp} X \mapsto 1 \& [p]; X\)

Finite approximations \(\rightarrow\) Practical implementation
Implementation and Case Studies

Several case studies

Interpreter in OCaml

Approximates the answer monotonically

Internet2’s Abilene backbone network
Conclusions

First language-based framework for specifying and verifying probabilistic network behavior.

van Wijngaarden would emphasize and promote the mathematical aspects of computing (...) in design principles of programming languages.

Analysis of several randomised routing protocols on real-world data.
Questions?
Analysed properties

(c) Max congestion

(d) Throughput

Values converge monotonically
Analysed properties

Failures

$$\ell_{1,2} \triangleq \text{sw}=S_1 ; \text{pt}=2 ; \text{dup} ; ((\text{sw} \leftarrow S_2 ; \text{pt}\leftarrow 1 ; \text{dup}) \oplus_{0.9} 0)$$

& $$\text{sw}=S_2 ; \text{pt}=1 ; \text{dup} ; ((\text{sw} \leftarrow S_1 ; \text{pt}\leftarrow 2 ; \text{dup}) \oplus_{0.9} 0)$$

(e) Max congestion

(f) Throughput
Routing

- Equal Cost Multipath Routing (ECMP)
- k-Shortest Paths (KSP)
- Multipath Routing (Multi)
- Oblivious Routing (Raecke)