Causal Reasoning in SDNs (NetKAT)

Georgiana Caltais, University of Konstanz
Shonan Seminar - “Causal Reasoning in Systems”
24-27 June, 2019
Outline

1. NetKAT - the Language
2. Reasoning & Verification
3. Towards a Framework for Causality

Sources:

“Programming, Modeling & Reasoning about Networks” (online tutorial by S.Smolka)

“NetKAT: Semantic Foundation for Networks” [C.J.Anderson et. al.], POPL’14

“A Fast Complier for NetKAT” [S.Smolka et. al.], ICFP’15
1. NetKAT - the Language

- Forwarding Along Paths
- Packet Classification
- Packet Modification

**KAT** [Kozen '96]

**NetKAT** ['14]

**Regular Expressions**
- +, ;, *

**Boolean Algebra**
- true, false, f=n, a&b, a|b, ¬a

**Network Primitives**
- f:=n, A→B
NetKAT Program - Example

\[\text{switch} = 6; \text{ port} = 88; \text{ dest} := 10.0.0.1; \]
\[(\text{port} := 50 + \text{port} := 51)\]

“For all packets incoming on port 88 of switch 6, set the destination IP address to 10.0.0.1 and multicast the packet out of ports 50 and 51.”
NetKAT Syntax & Semantics

Syntax

Fields  \[ f ::= f_1 \mid \cdots \mid f_k \]

Packets  \[ pk ::= \{f_1 = v_1, \cdots, f_k = v_k\} \]

Histories  \[ h ::= pk::\langle \mid pk::h \]

Predicates  \[ a, b ::= 1 \mid 0 \mid f = n \mid a + b \mid a \cdot b \mid \neg a \]

Identity  Drop  Test  Disjunction  Conjunction  Negation

Policies  \[ p, q ::= a \mid f \leftarrow n \mid p + q \mid p \cdot q \]

Filter  Modification  Union  Sequential composition  Kleene star  Duplication

Semantics

\[ [p] \in H \rightarrow \mathcal{P}(H) \]

\[ [1] h \triangleq \{h\} \]

\[ [0] h \triangleq \{\} \]

\[ [f = n] (pk::h) \triangleq \begin{cases} \{pk::h\} & \text{if } pk.f = n \\ \{\} & \text{otherwise} \end{cases} \]

\[ \neg [a] h \triangleq \{h\} \setminus ([a] h) \]

\[ [f \leftarrow n] (pk::h) \triangleq \{pk[f := n]::h\} \]

\[ [p + q] h \triangleq [p] h \cup [q] h \]

\[ [p \cdot q] h \triangleq ([p] \cdot [q]) h \]

\[ [p^*] h \triangleq \bigcup_{i \in \mathbb{N}} F^i h \]

where \( F^0 h \triangleq \{h\} \) and \( F^{i+1} h \triangleq ([p] \cdot F^i) h \)

\[ [\text{dup}] (pk::h) \triangleq \{pk::(pk::h)\} \]
Encoding Switch Forwarding Tables

\begin{align*}
\text{Pattern} & \quad \text{Action} & \quad pol_A & \triangleq pt \leftarrow 2 \\
* & \quad pt \leftarrow 2 & & \\
\text{Pattern} & \quad \text{Action} & \quad pol_B & \triangleq dst = A \\
dst = A & \quad \text{true} & & \\
* & & & \\
\text{Pattern} & \quad \text{Action} & \quad pol_B \cdot pol_A & \\
dst = A & \quad pt \leftarrow 2 & & false \\
* & & & \\
\text{(a)} & \text{An atomic modification} & & \\
\text{Pattern} & \quad \text{Action} & \quad pol_D & \triangleq dst = A \cdot pt \leftarrow 1 + \\
dst = A & \quad pt \leftarrow 1 & & dst = B \cdot pt \leftarrow 2 \\
dst = B & \quad pt \leftarrow 2 & & false \\
* & & & \\
\text{Pattern} & \quad \text{Action} & \quad pol_E & \triangleq \left( \text{proto} = \text{ssh} \right) + pol_A \\
dst = A & \quad pt \leftarrow 3 & & false \\
proto = \text{ssh} & \quad pt \leftarrow 3 & & \\
* & & & \\
\text{(b)} & \text{An atomic predicate} & & \\
\text{(c)} & \text{Forwarding to a single host} & & \\
\text{Pattern} & \quad \text{Action} & \quad pol_E & \triangleq \left( \text{proto} = \text{ssh} \right) \cdot pt \leftarrow 3 \\
dst = A & \quad pt \leftarrow 3 & & \\
proto = \text{ssh} & \quad pt \leftarrow 3 & & \\
* & & & \\
\text{(d)} & \text{Forwarding traffic to two hosts} & & \\
\text{(e)} & \text{Monitoring SSH traffic and traffic to host A} & & \\
\end{align*}
Encoding Network Topologies (I)

topology: e.g.

\[ t = \text{sw} = A \cdot \text{pt} = 5 \cdot \text{sw} = B \cdot \text{pt} = 6 \cdot \text{sw} = A \cdot (\text{pt} = 1 + \text{pt} = 3) + \text{sw} = B \cdot (\text{pt} = 2 + \text{pt} = 4) \]
Encoding Network Topologies (II)

[Diagram of network topology]

topology (assuming all ports different):

\[ t = pt = 5 \land pt = 6 \lor pt = 5 \lor pt = 1 \lor pt = 2 \lor pt = 3 \lor pt = 4 \]
Encoding Networks

A network can be encoded in NetKAT by interleaving steps of processing by switches and topology.
Encoding Networks

A network can be encoded in NetKAT by interleaving steps of processing by switches and topology

(topology; switch)*
Encoding Networks

A network can be encoded in NetKAT by interleaving steps of processing by switches and topology.

(topology; switch)*
Encoding Networks

A network can be encoded in NetKAT by interleaving steps of processing by switches and topology.
Encoding Networks

A network can be encoded in NetKAT by interleaving steps of processing by switches and topology.

\[(\text{topology}; \text{switch})^*\]
2. Reasoning & Verification
Network Verification

- Sound & Complete Axiomatisation [C.J.Anderson et. al.]

Kleene Algebra Axioms

\[
\begin{align*}
    p + (q + r) & \equiv (p + q) + r \\
    p + q & \equiv q + p \\
    p + 0 & \equiv p \\
    p + p & \equiv p \\
    p \cdot (q \cdot r) & \equiv (p \cdot q) \cdot r \\
    1 \cdot p & \equiv p \\
    p \cdot 1 & \equiv p \\
    p \cdot (q + r) & \equiv p \cdot q + p \cdot r \\
    (p + q) \cdot r & \equiv p \cdot r + q \cdot r \\
    0 \cdot p & \equiv 0 \\
    p \cdot 0 & \equiv 0 \\
    1 + p \cdot p^* & \equiv p^* \\
    q + p \cdot r & \leq r \Rightarrow p^* \cdot q \leq r \\
    1 + p^* \cdot p & \equiv p^* \\
    p + q \cdot r & \leq q \Rightarrow p \cdot r^* \leq q
\end{align*}
\]

KA-PLUS-ASSOC
KA-PLUS-COMM
KA-PLUS-ZERO
KA-PLUS-IDEM
KA-SEQ-ASSOC
KA-ONE-SEQ
KA-SEQ-ONE
KA-SEQ-DIST-L
KA-SEQ-DIST-R
KA-ZERO-SEQ
KA-SEQ-ZERO
KA-UNROLL-L
KA-LFP-L
KA-UNROLL-R
KA-LFP-R
Network Verification

- Sound & Complete Axiomatisation [C.J.Anderson et. al.]

**Additional Boolean Algebra Axioms**

\[
\begin{align*}
    a + (b \cdot c) & \equiv (a + b) \cdot (a + c) \\
    a + 1 & \equiv 1 \\
    a + \neg a & \equiv 1 \\
    a \cdot b & \equiv b \cdot a \\
    a \cdot \neg a & \equiv 0 \\
    a \cdot a & \equiv a
\end{align*}
\]

- BA-PLUS-DIST
- BA-PLUS-ONE
- BA-EXCL-MID
- BA-SEQ-COMM
- BA-CONTRA
- BA-SEQ-IDEM
Network Verification

• Sound & Complete Axiomatisation [C.J.Anderson et. al.]

Packet Algebra Axioms

\[
\begin{align*}
  f & \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n, \text{ if } f \neq f' \\
  f & \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n, \text{ if } f \neq f' \\
  \text{dup} \cdot f = n & \equiv f = n \cdot \text{dup} \\
  f & \leftarrow n \cdot f = n \equiv f \leftarrow n \\
  f & = n \cdot f \leftarrow n \equiv f = n \\
  f & \leftarrow n \cdot f \leftarrow n' \equiv f \leftarrow n' \\
  f = n \cdot f = n' & \equiv 0, \text{ if } n \neq n' \\
  \sum_i f = i & \equiv 1
\end{align*}
\]

PA-MOD-MOD-COMM
PA-MOD-FILTER-COMM
PA-DUP-FILTER-COMM
PA-MOD-FILTER
PA-FILTER-MOD
PA-MOD-MOD
PA-CONTRA
PA-MATCH-ALL
Network Verification

- **Sound & Complete** Axiomatisation [C.J.Anderson et. al.]

\[
[[p]] = [[q]] \iff \lnot p = q
\]

- E.g., **Reachability**:

“Does the network forward from ingress (in) to egress (out)?”

NO \iff \lnot \text{in . (switch.topology)* . out} = 0

YES \iff \lnot \text{in . (switch.topology)* . out} =/= 0
Reasoning About Correctness of NetKAT Programs

- Programmer 1 has to implement a switch policy s.t.:
  
  “H1 can only forward to H2”

- Correctness:
  
  - H1 can forward to H2 (H1 $\rightarrow\rightarrow$ H2)
  
  - H1 cannot forward to H3 or H4 (H1 $\neg\neg\rightarrow\rightarrow$ H3,4)
Reasoning About Correctness of NetKAT Programs

“H1 can only forward to H2”

- Policy \( p_1 : (pt = 1 \implies pt \leftarrow 5) + (pt = 6 \implies pt \leftarrow 2) \)

H1 can forward to H2 (H1 \( \rightarrow\rightarrow\) H2)

- \( \vdash (pt = 1) \cdot (p_1 \cdot t)^* \cdot (pt = 2) =/\neq 0 \)

H1 cannot forward to H3 or H4 (H1 \( \rightarrow\rightarrow\rightarrow\) H3,4)

- \( \vdash (pt = 1) \cdot (p_1 \cdot t)^* \cdot (pt = 3 + pt = 4) = 0 \)
Reasoning About Correctness of NetKAT Programs

- **Programmer 2** has to implement a switch policy s.t.:

  “H3 can only forward to H4”

- **Correctness:**
  
  - H3 can forward to H4 (H3 —>> H4)
  
  - H3 cannot forward to H1 or H2 (H3 -/->> H1,2)
Reasoning About Correctness of NetKAT Programs

- Programmer 1: “H1 can only forward to H2” / switch policy \( p_1 \)
- Programmer 2: “H3 can only forward to H4” / switch policy \( p_2 \)
- Assume Programmer 3 implements \( p \) as the union of the two correct policies \( p_1 \) and \( p_2 \)

\[
p = p_1 + p_2
\]

- Network becomes \((p \cdot t)^* = ((p_1 + p_2) \cdot t)^*\)
- Does H1 \(-/->>\) H3,4 still hold?
Reasoning About Correctness of NetKAT Programs

H1 -/-:> H3,4 holds iff

\[ \vdash \text{pt} = 1 \cdot ((p_1 + p_2) \cdot t)^* \cdot (\text{pt} = 3 + \text{pt} = 4) = 0 \text{ iff} \]

(acc. to NetKAT axioms)

\[ \vdash \text{pt} = 1 \cdot \text{pt} \leftarrow 4 + P = 0 \]

What is the cause?
3. Towards a Framework for Causality
What Is the Cause?  
- Obvious Challenges -

H1 \not\implies H3,4 holds iff

\[ \vdash pt = 1 \cdot ((p1 + p2) \cdot t)^* \cdot (pt = 3 + pt = 4) = 0 \text{ iff} \]

(acc. to NetKAT axioms)

\[ \vdash pt = 1 \cdot pt \leftarrow 4 + P = 0 \]

provides too little information

contains *
What Is the Cause?  
- Obvious Challenges -

H1 -/->> H3,4 holds iff

\[ \vdash pt = 1 \cdot ((p1 + p2) \cdot t)^* \cdot (pt = 3 + pt = 4) = 0 \text{ iff} \]

(acc. to NetKAT axioms)

\[ \vdash pt = 1 \cdot pt \leftarrow 4 + P = 0 \]

provides too little information

“Star Elimination” in [C.J.Anderson et. al]
assumption: no dup, no sw <-
uses all axioms to build the Normal Form of P, NF (P)
\[ \vdash P \sim NF(P) \]
... provides too little information as well...
What Is the Cause?
- Possible Solution -

\[ \neg pt = 1 \cdot ((p1 + p2) \cdot t)^* \cdot (pt = 3 + pt = 4) = 0 \text{ iff } (\ldots \text{ axioms}) \]

\[ \neg pt = 1 \cdot pt \leftarrow 1 \cdot pt \leftarrow 5 \cdot pt \leftarrow 6 \cdot pt \leftarrow 4 + P_{sf} = 0 \]

Inhibit some of the axioms, e.g.:
\[ f \leftarrow n \cdot f \leftarrow n' = f \leftarrow n' \text{ [PA-MOD-MOD]} \]

“Approximate” \[ (p.t)^* = (1 + p.t)^n \text{ for some } n \ldots \]

and remove *-unfolding axioms
* “Approximation”

A - the set of all tests \( f \in \mathcal{A} \)
\( \pi \) - the set of all assignments \( f \in \pi \)

Assume \( p = \sum_{i=1}^{n} \alpha_i \cdot \pi_i \), with \( \alpha_i \in \mathbb{A}^x \), \( \pi_i \in \pi^x \)

\( t \) - a topology

Observation: \( p \) entails loop-free path from in to out crossing at most \( n \) switches

\[ (p \cdot t)^* \equiv (1 + p \cdot t)^n \]

⇒ new axiom
Some Terminology...

Let $\vdash_*$ be the entailment relation over the new axiomatization

$p \in \text{Tree-Form} \iff p = \prod_{i=1}^{n} p_i \mid p_i \in (\mathcal{A}^*, \Pi_*)$

notation: $p_i \in P \Rightarrow T \vdash_*(p)$

$\text{support}(g) = \{ \exists x \in g \mid \exists \tilde{y} \in \mathcal{A} \mid \tilde{y} \subset \tilde{y}, \tilde{y} \subseteq \tilde{y} \}$, $g \in \text{Tree-Form}$
Consider the **SAFETY** property:
\[
T \text{ im.}(p.t)^\ast \cdot \text{out} \equiv 0 \quad (\star)
\]

The **CAUSE** w.r.t. the violation of (\star) is
\[
C = \sum_{\overline{2} \in \text{support}(2')} \overline{2} \quad \text{where}
\]
\[
T \ast \text{im.}(p.t)^\ast \cdot \text{out} \equiv 2, \quad 2 \neq 0
\]
\[
2' = T \oplus (2)
\]

**Conjecture**: \( T \cdot C \equiv N \oplus (\text{im.}(p.t)^\ast \cdot \text{out}) \)
Questions?

• Current & Future Work:
  • Trace back the cause into the original code
  • How does the counterfactual look like?
  • Handling other interesting network properties
    • E.g., waypointing…
  • Responsibility, blame