Indirect Network Troubleshooting with The Chase

Mubashir Anwar†, Fangping Lan*, Anduo Wang*, Matthew Caesar†
†*University of Illinois Urbana-Campaign*, **Temple University*

Two clouds obscuring network verification

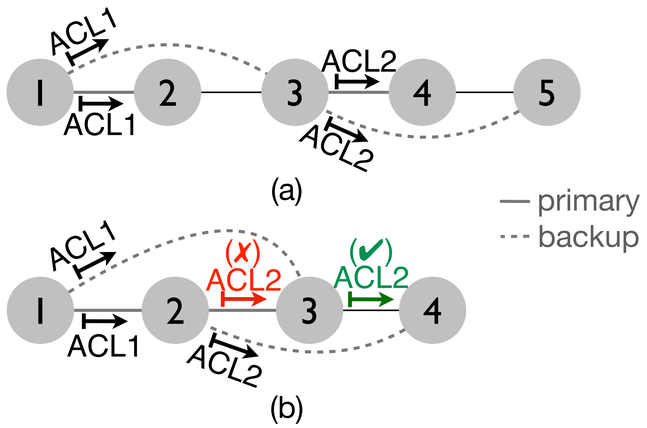
- Cloud I: Concurrent Events
  - failures and changes are common in networks
- Cloud II: Distributed Policies
  - global invariants are ensured using policies in different parts of the network
- Problem Formulation:
  - I: Temporal Decision Problem: Does a policy still hold under different network environments (e.g. failures)?
  - II: Spatial Decision Problem: Do local policies (under different network partitions) imply a network-wide invariant?

Indirect Troubleshooting

- Solves an implication problem
  - decides if known facts about the network imply some unknown property
  - natively supports reasoning about concurrent events (link failures) and distributed policies
  - a unifying implementation with *the chase*
- *The chase:* Tests implications among data dependencies in databases
  - a unifying database primitive (denoted, chase(γ, σ)) denotes the impact of applying the premise σ to the conclusion γ
  - premise and conclusion represented as tableaux
  - premise could be policies verified by existing verification tools directly

Cloud I: Concurrent Events

Goal: Multi-path consistency



Challenge: Verifying individual events is not enough to verify their combined effect

Modeling and Application

Network policy as a tableau

γ_c	F	N	A
T_c	x_f	1	∅
u_c	x_f	5	$\{a_1, a_2\}$

required application of ACLs on the network

Data dependency on a failure as a tableau

σ_{c12}	F	N	A
	x_f	1	x_a
	x_f	3	$x_a \cup \{a_1\}$

application of ACL for failure event on link 1-2

Application of a dependency

	F	N	A
	x_f	1	∅
	x_f	3	$\{a_1\}$
	x_f	5	$\{a_1, a_2\}$

applying σ_{c12} to γ_c

Final result

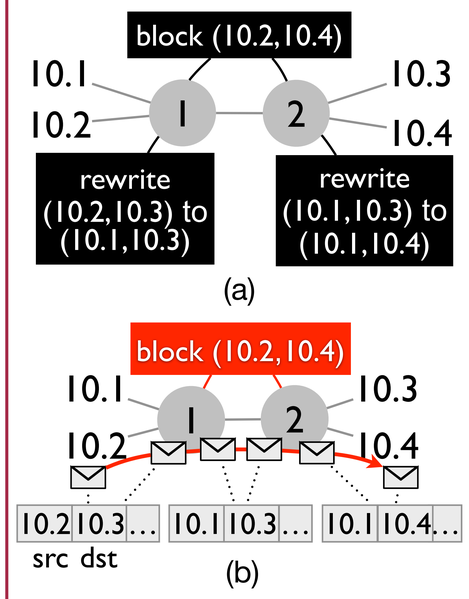
γ_c'	F	N	A
	x_f	1	∅
T_c'	x_f	3	$\{a_1\}$
	x_f	5	$\{a_1, a_2\}$
u_c	x_f	5	$\{a_1, a_2\}$

applying all dependencies to γ_c

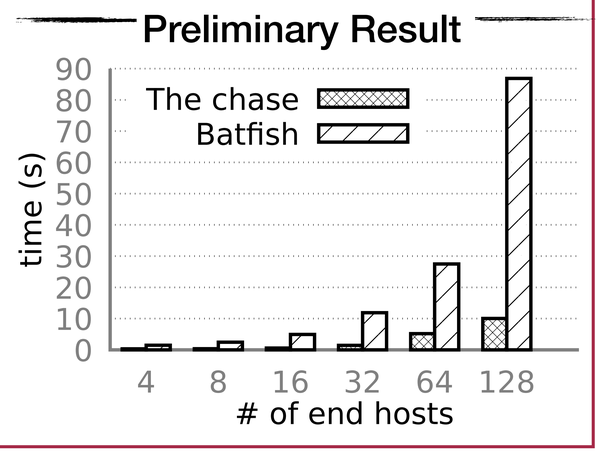
This work is supported by the NSF Awards CNS-1909450 and CNS-2145242

Cloud II: Distributed Policies

Goal: Block traffic from 10.2 to 10.4



Challenge: Two correct partitions, when combined, do not maintain a network invariant

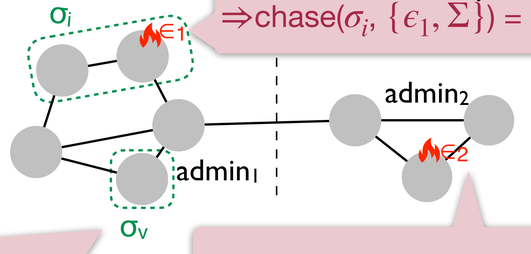


Future Possibilities

- Networks are hard to reason about
  - grow without a premeditated plan
  - different parts often managed by different departments
- *The chase* provides a flexible framework to reason with disparate network views

is policy σ_i unaffected when a new error ϵ_1 is detected?
⇒chase(σ_i, $\{\epsilon_1, \Sigma\}$) = σ_i?

is detected failure ϵ_1 a sure symptom of σ_i?
⇒chase(σ_i, $\{\epsilon_1, \Sigma\}$) = ⊥?



is new policy σ_v redundant (impossible)?
⇒chase(σ_v, Σ) = ⊤(⊥)?

is remote ϵ_2 bound to occur if ϵ_1 is detected?
⇒chase(ϵ_2, $\{\epsilon_1, \Sigma\}$) = ⊤?