

# NetSure: A Continuous Way to Prove Your Network Behaves Correctly

Sung Woo Jeon<sup>+</sup>, Arjun Gama<sup>+</sup>, Tanya Banerjee<sup>\*</sup>, Wadia Ganim<sup>\*</sup>, Kulani Mahadewa<sup>\*</sup>, Matthew Caesar<sup>+</sup>  
<sup>\*</sup>Illinois Advanced Research Center at Singapore Ltd., <sup>+</sup>University of Illinois at Urbana-Champaign

## Motivation

- Modern cyber infrastructures are **highly interconnected** and **programmable**, where even minor misconfigurations can silently escalate into outages or security incidents.
- Existing network-wide verification relies on configuration-based abstract models, while individual network program verification misses emergent, system-wide behavior.
- In dynamic network environment, actual runtime behavior can diverge from what is verified, leaving correctness assumed rather than continuously assured.

**Vision: What if network misconfigurations and security violations could never go unnoticed?**

## Our Solution

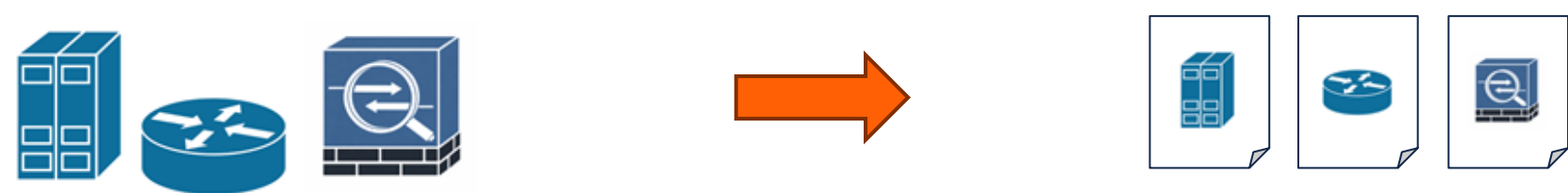
NetSure enables continuous behavioral verification in two core stages.

- Network-to-Program Translation:** First translate the observed network behavior into an executable model
- Behavioral Verification:** Apply program analysis techniques on the model to check network-wide invariants.

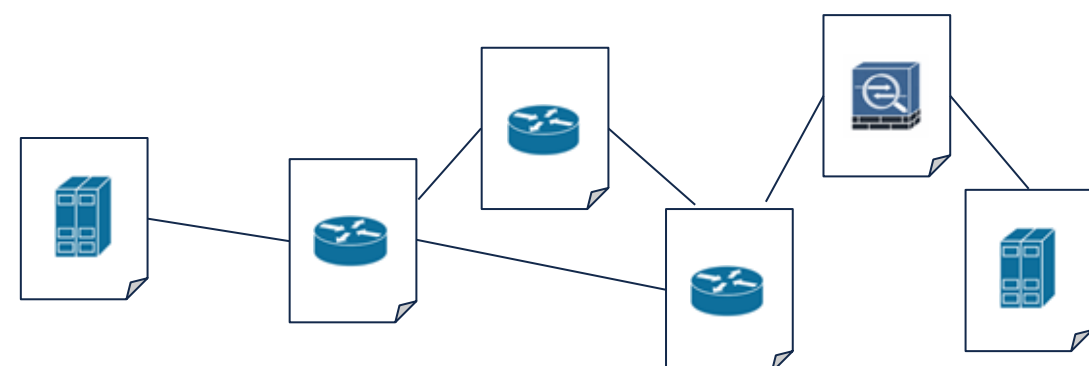
### 1) Network-to-Program Translation:

Observed packet behaviors are transformed into an executable network program in two steps.

- Device Model Extraction:** Learn individual device behavior from ingress-egress packet observations.



- End-to-End Composition:** Combine device behaviors into a unified network execution model.



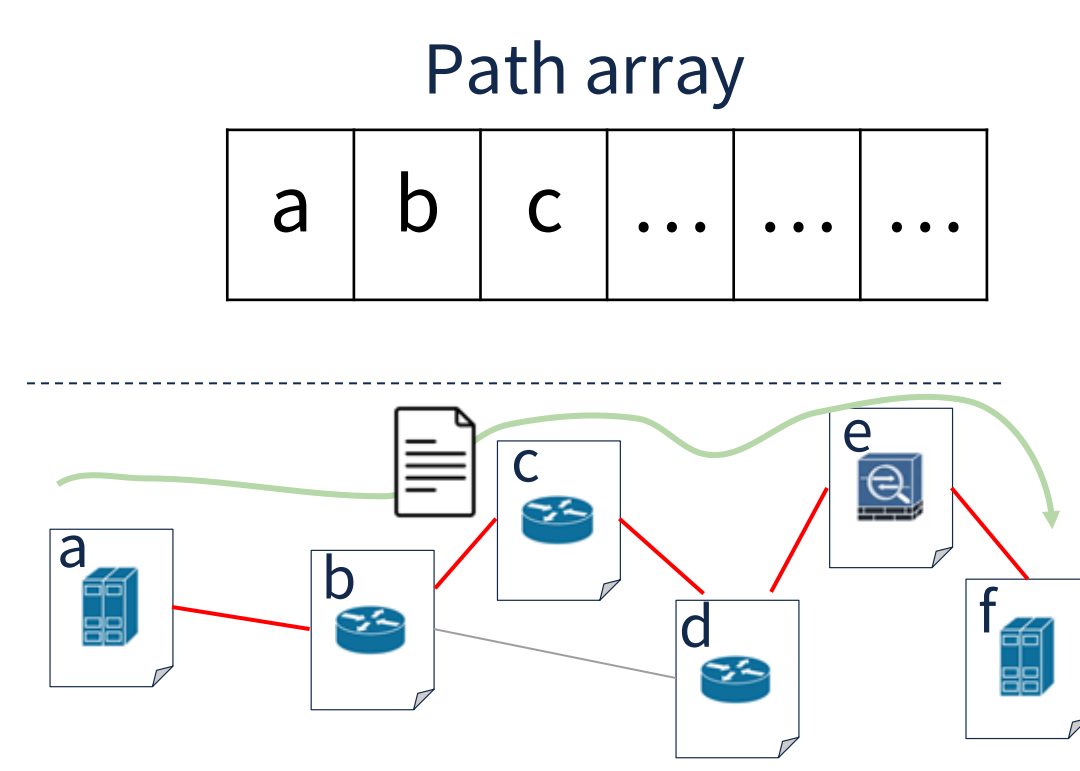
### 2) Behavioral Verification:

The synthesized executable network model is instrumented and analyzed to detect violations of **network-wide invariants** over observed behavior.

Class	Focus	Example Invariants
Basic	Structural packet behaviour	Reachability, Isolation, Waypoint, Loop Freedom
Statistical	Traffic pattern consistency	Load Balancing, Path Length Bound, Path Entropy
Application-level	Service and policy behaviour	NetCache Consistency, IoT Control Correctness, Security Enforcement

### Methodology:

- Instrumenting the synthesized executable model (e.g., logging variables, packet fields, paths, counters).
- Running program-analysis style checks over executions. (e.g., Daikon Invariant Checker)



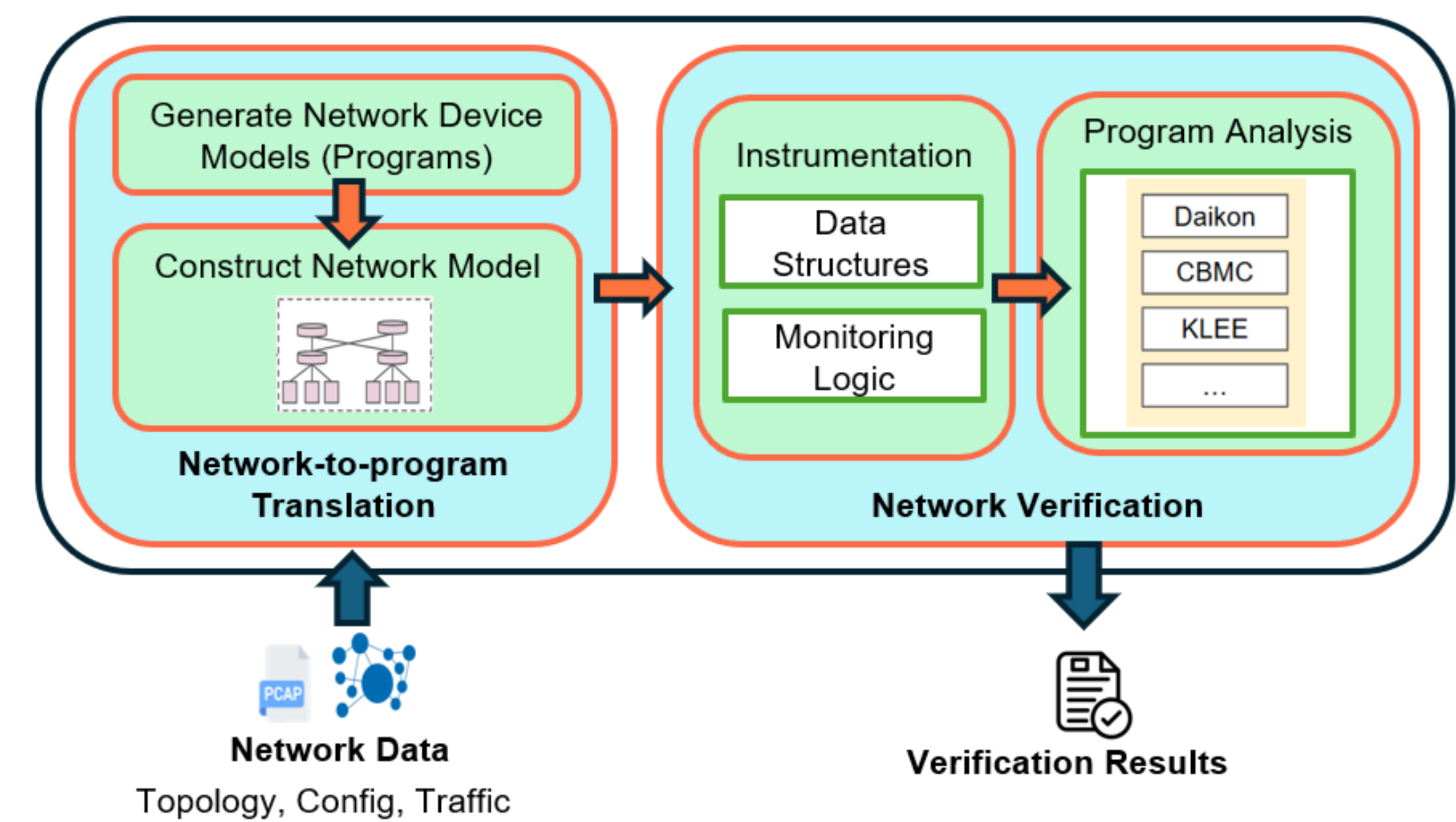
Do packets always visit c? → Does the array always contain c?

## Objectives

Our goal is to **design behavioral verification framework** that:

- Transforms real network behavior into executable models (software) for rigorous reasoning.
- Explores the use of **program analysis** techniques on the extracted models to verify **network-wide invariants** over observed behavior.

## System Architecture



## NetSure Demo

Category	Invariant	Examine	Verified
Basic	Reachability	<input type="checkbox"/>	
	Isolation	<input type="checkbox"/>	
	Waypoint	<input type="checkbox"/>	
Statistical	Loop Freedom	<input checked="" type="checkbox"/>	Yes
	Load Balancing	<input type="checkbox"/>	
	Path Length Bound	<input checked="" type="checkbox"/>	Yes
	Path Entropy	<input checked="" type="checkbox"/>	No

## Evaluation

Evaluation results show that NetSure scales linearly with program synthesis and converges quickly, demonstrating efficient runtime performance and manageable model growth.

