**Problem Statement and Goals**

**Motivation:** networks require near-constant configuration changes [1]
- 20% of network operators make changes once per day
- 80% of network operators are concerned changes will introduce problems with existing functionality
- Operators need a way to vet changes at a high level

**Goals:**
- Mine succinct summaries of configuration changes
- Understand low-level configuration changes: infer high-level intention
- Verify operational updates: confirm compliance with intention and network policy

**Path Change Summaries:**
A configuration change can encompass many tasks (re-routing traffic, updating ACLs, modifying interface/port settings). Initially, we focus on *path changes* and summarize each change in the form:

```
p: old_path => new_path
```

- **pc:** a packet class, an equivalence class where every packet is forward the same way [3]
- **old_path, new_path:** regular expressions defining a path in the previous network and the current network, respectively

**Generalizing Useful Path Expressions**

**Key Challenge:** deriving a regular expression that describes the path change at the right level of abstraction
- **Precise:** informative enough to capture the impact of the configuration change
  - new_path: .* - not precise enough to describe impact
- **Concise:** uncover the high-level intention of the configuration change
  - old_path: .* - concisely matches all previous paths

**Mining Strategies:**
- **Correctness:** the expression correctly identifies the change and could be used to synthesize a change [2]
- **Minimality:** bias toward expressions with fewer terms (Occam’s razor)
- **Topology restrictions:** if only a single path exists between nodes $n_1$ and $n_2$, ignore intermediate hops
- **Non-empty path change:** the difference between old_path and new_path is non-empty
- **Indistinguishable nodes:** automatically inferred or user-defined sets of nodes with similar function

**Motivating Example**

*Input:* two network configurations: $N \rightarrow N'$
*Output:* summary of each changed path, as a regular expression

- The most generic expression does not capture the intention of the configuration change:  
  .* => .*
- An explicit expression is too verbose: 
  (A+B+C) F1 X Z => (A+B+C) F2 Y Z
- Goal: a concise, useful expression: 
  .* F1 .* => .* F2 .*

**Application of Mining Strategies**

**Indistinguishable nodes:** automatically infer and cluster together devices with similar functionality
- $N = \text{all nodes in the network}$
- Set of firewalls: $fw = \{fw_1, fw_2\}$
- Set of non-firewalls: $nf = N - fw$

- Summarized path: src $nf* fw nf* dst$

**Topology restrictions:**

- Summarized path: src .* dst

**References**