University of Mons
Sciences Faculty
Computer Science Department

Probabilistic Verification of Network Configuration

Explanatory Report

Supervisor :
Bruno Quoitin

Author :
Maxime Bartha

Academic Year 2025-2026

Contents

1 Intro

Table 1: Properties supported by NetDice

	Property	Meaning
Single-Flow	$Reach_{u \to d}$	Reachability from source u to destination d
	$Waypoint_{u \to d}(w)$	Traffic from u to d traverses w
	$Egress_{u \to d}(e)$	Traffic from u to d leaves the network at e
	$PathLenght_{u \to d}(l)$	Traffic from u to d traverses l links
Mutli-Flow	$Balanced_F(\mathcal{L}, \Delta)$	Under flows F with given volumes, the load on the links in $\mathcal{L}$ differs by at most Δ
	$Isolation_F$	Set of flows in F doesn't share any links
	$Congestion_F(l, t)$	Together, all flows in F with their volume don't exceed threshold t for link l

2 Probabilistic Verification of Network Configuration

In this section, we explore the reasoning and implementation behind NetDice [1], a probabilistic verification tool for networks configuration.

It is able to verify properties on single or multiples flows with respect to certain nodes or links in a network. A flow in the context of internet routing is generally defined by a source and destination IP address, ports numbers and a transport protocol. We use a simplification considering the source and destination IP addresses. Examples of properties and their meaning are represented in Table 1.

In NetDice, equal-cost multiple path routing (ECMP) is considered, hence a flow in a network can take multiples paths if they share the same cost. To represent such behavior, forwarding graphs, composed of V_{fwd} a set of vertices (routers) and E_{fwd} a set of directed edges (links), are used. Those forwarding graphs can be altered by link failures represented by a failure scenario which is a set of unusable links in the network configuration.

To compute the probability of a certain failure scenario, two mathematical tools are used :

1. A vector L of random variables $s.t.$ $L_e \in \{0, 1\}$ for each link e to represent if the link is up ($L_e = 1$) or down ($L_e = 0$)

2. A probability mass function of link failure $P(L)$

Such that, for a failure scenario represented by a vector l, its probability is defined by $P(L = l)$. This probability mass function is joint meaning it can represent complex relation between link failures, such as node failure (all incident links) or shared risks link group (SLRG) which are discussed in section 2.2.

Given a network configuration, a property ϕ and a link failure distribution L, the goal of NetDice is to compute the probability of the property being satisfied.

A naive approach could be to compute all possible failure scenarios and sum the probabilities of the scenarios for which the property holds. But the key insight of NetDice is to prune failure scenarios where links are guaranteed to have no impact on the state of ϕ. To this end, NetDice introduces the concept of cold edge.

Definition 1 *Given a network configuration, a property ϕ and a flow (u, d), a subset of edges $\mathcal{C}$ is cold iff any combination of failures in $\mathcal{C}$ is guaranteed not to change the forwarding graph for (u, d). An edge is called cold if it belongs to $\mathcal{C}$ and hot otherwise.*

In practice, hot edges are used to prune the exploration of the possible failure scenarios. Intuitively, as they impact the forwarding graph, we only explore failure scenarios where hot edges fail to avoid redundant scenarios due to the unchanged forwarding graphs.

An algorithm to compute hot edges considering the shortest paths and static routes is presented in Alg.1. It takes a flow (u, d), a forwarding graph's edges E_{fwd}, a set of static routes $Static_d$ and a failure scenario l.
All edges along the shortest path and the traversed static routes are hot as a failure in any of them would change if ϕ holds due to unreachability or changes in the forwarding graph.
To this end, every edge along the shortest path from the source u to the destination d is considered hot (Lin.2). As the shortest path is considered by u, any failure in this shortest path will impact the decision of the source node hence the resulting forwarding graph.
Then edges along the shortest path from every end of a traversed static route are added as hot (Lin.4). As those static routes deviate the forwarding graph and the end node of a static route considers shortest path routing.
Finally, every edge on traversed static route is added as hot (Lin.5).

Algorithm 1 Hot edges for static routes and shortest path

procedure HOTSPSTATIC(u, d, E_{fwd}, l)
  $\mathcal{H} \leftarrow AllSP(\{u\}, \{d\}, l)$ ▷ initial shortest path
  $\mathcal{S} \leftarrow \{y \mid (x, y) \in Static_d \cap E_{fwd}\}$ ▷ end nodes of static route
  $\mathcal{H} \leftarrow \mathcal{H} \cup AllSP(\mathcal{S}, \{d\}, l)$
  return $\mathcal{H} \cup (Static_d \cap E_{fwd})$

procedure ALLSP($\mathcal{S}, \mathcal{T}, l$)
  return $\bigcup_{s \in \mathcal{S}, t \in \mathcal{T}} SP_l(s, t)$

As an example, Figure 1 represents the hot and cold edges of a certain network configuration considering no failure and a flow (D, C), the forwarding edges $E_fwd = (D - E), (E - B), (B - C)$ and the static routes $Static_d = (E - B)$. Note that the E - C edge is marked as hot even though it is not in the forwarding graph because it lies on the shortest path of D. If the E-C link fails, the decision of D would be to use D-A-B-C as its shortest path, changing the forwarding graph.

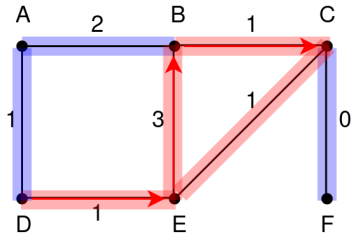

Figure 1: Cold Edges in a network configuration with no failures and a flow (D, C). E - B is a static route. Blue and red highlight show cold and hot edges. The forwarding graph is represented by the red arrows

Computing P(ϕ), intuitively, is done by only exploring possible failures of hot edges. A simple probability function mapping each edge to a certain failure probability is considered here, more complex probability failure model are discussed in 2.2. Moreover, a full algorithm incorporating BGP is presented in Section 2.1. To explore the failure space, NetDice considers 3 possible states for an edge during the exploration process :

1. down : the edge cannot be traversed in current and future scenarios
2. up : the edge can be traversed in current and future scenarios

3. undefined : the edge is currently traversable but might not be in the future

Those states are only considered during the exploration process. In terms of forwarding graph computation, within a certain scenario, down edges are not considered usable whereas up and undefined are considered usable. It is assumed that a *check* function returning, given a property and forwarding graphs, if the property holds in every forwarding graph or not. In practice, it is implemented using variants of DFS depending on the property to check.

The exploration procedure can be described as follows :

0. initially, all edges are undefined (all traversable)
1. compute the forwarding graphs with respect to the failure scenario
2. compute the hot edges with ϕ and the forwarding graphs
3. let $\mathcal{H}^?$ be the set of undefined hot edges
4. $\forall h \in \mathcal{H}^?$ make two recursive calls (going back to .1) : with h up and h down
5. return the sum of the recursive calls and the probability of the actual scenario if ϕ holds in it

Recall that we, multiple forwarding graphs are possibly computed as equal-cost multi path (ECMP) is considered.

An example of execution on the network configuration from Figure 1 with property ϕ "Traffic from D to C traverses E" is shown at Figure 2. Only the exploration is shown, the probability of the properties could be computed easily considering marginal properties for edges to fail. As required, initially all edges are undefined, the hot edges set is computed and recursive calls are made for every hot edge. At the first recursive call (D $\neq$ E), we remark that its first recursive call on (A - D) results in a scenario where C isn't reachable from D, meaning every edge is cold. This lead to a dead end in the exploration as no more hot edge is to mark as down. A similar situation happens in the last recursive calls, except it's due to a loop in the forwarding graph because E uses its static route and B tries to reach C by E as (B - C) is down.

In the example, only 16 failure scenarios are explored. In contrast to the $2^7 = 128$ scenarios a Brute Force algorithm would have had to explore.

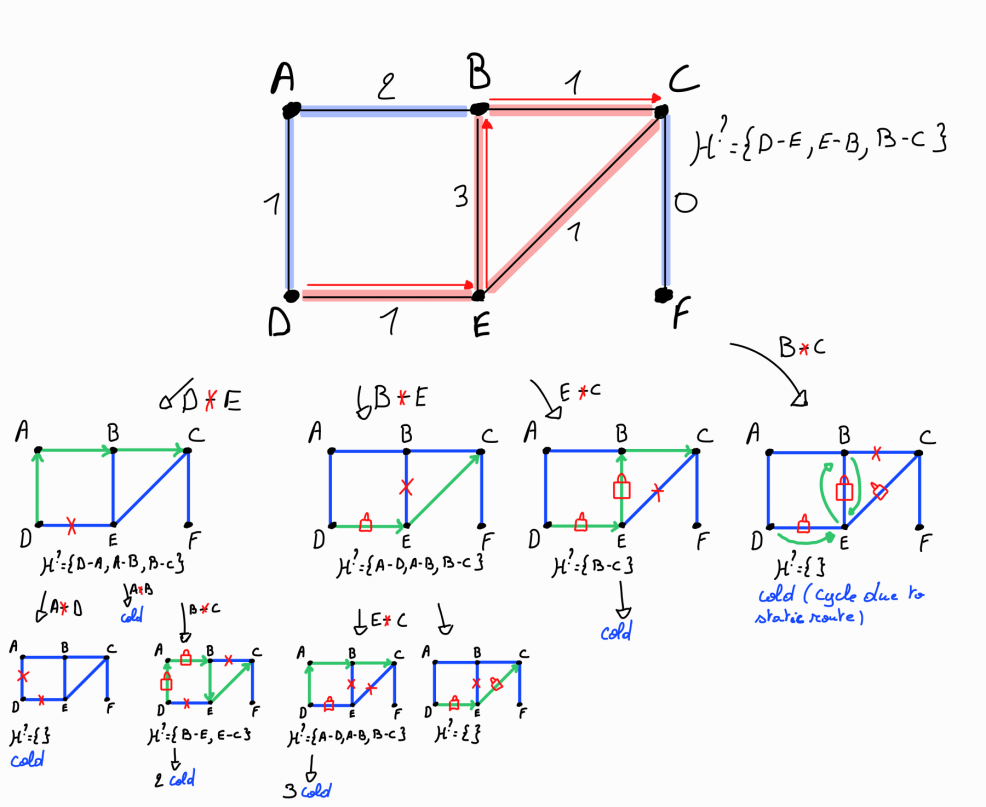

Figure 2: Exploration of the failure space with ϕ"Traffic from D to C traverses E". Crosses represents down links, padlocks represents up links. Absence of symbol represents undefined links. "Cold" in blue represents scenarios containing only cold edges (which stops the exploration).

2.1 Incorporating BGP

2.2 Complex Link Failure Probability model

2.3 Results

References

[1] Samuel Steffen et al. “Probabilistic Verification of Network Configurations”. In: *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*. SIGCOMM ’20. Virtual Event, USA: Association for Computing Machinery, 2020,

Algorithm 2 Failures exploration to compute $P(\phi)$ considering static routes and shortest path

1: **procedure** EXPLORE(s)
2: $L \leftarrow fill(s)$, $\mathcal{H} \leftarrow \emptyset$, $\rho \leftarrow 0$
3: **for** each flow $(u_i, d_i) \in flows(\phi)$ **do**
4: compute forwarding graph (V^i_{fwd}, E^i_{fwd}) for (u_i, d_i) under L
5: $\mathcal{H} \leftarrow \mathcal{H} \cup HotSpStatic(u_i, d_i, E^i_{fwd}, L)$
6: $\mathcal{H} \leftarrow \mathcal{H} \cup \{e | e \in E \wedge s(e) = ?\}$
7: **if** Check(ϕ, $(V^1_{fwd}, E^1_{fwd}), \ldots, (V^n_{fwd}, E^n_{fwd})$) **then**
8: $\rho \leftarrow \rho + prob(s[\mathcal{H}^? \leftarrow 1])$
9: $s' \leftarrow s$
10: **for** $l \in \mathcal{H}^?$ **do**
11: $s' \leftarrow s'[l \leftarrow 0]$
12: $\rho \leftarrow \rho + Explore(s')$
13: $s' \leftarrow s'[l \leftarrow 1]$
14: **return** ρ

pp. 750–764. ISBN: 9781450379557. DOI: 10.1145/3387514.3405900. URL: https://doi.org/10.1145/3387514.3405900.