Control plane compression for fault tolerance

Nick Giannarakis*
PhD Candidate, ACM number: 4185942
Princeton University
nick.giannarakis@princeton.edu

Network Verification: A scaling challenge

Routing decisions in modern networks are often made by distributed protocols such as BGP or OSPF. Each router in the network executes one or more protocols in order to select a route and communicate it to its neighbors. Protocols are parameterized by a per-router configuration that affects their execution, allowing network operators to enforce a variety of end-to-end routing policies.

This flexibility however comes at a cost. Configuring each router to achieve the desired behavior of the distributed system is a difficult task[8, 10, 12]. And the problem is further amplified by *failures* in the network; operators have to ensure the sensible behavior of the network even when some link or router stops working.

A wide range of network verification [3, 7, 13] and simulation [5, 6, 11] techniques have been suggested in order to automate reasoning about the behavior of a network. However, most of those techniques — especially the ones that make soundness and/or completeness guarantees — don't scale to the size of real networks.

Background: Network compression

In order to tackle the large scale of modern networks, Beckett *et al.* developed *Bonsai* [4], a theory of equivalence between networks and an efficient algorithm that given a *concrete* network constructs an equivalent *abstract* network of a smaller size and thus amenable to verification.

Intuitively, abstract nodes can be seen as sets of concrete nodes that share a *similar* forwarding behavior. For example aggregation routers in a fat tree network [1] usually have similar configurations and connectivity and hence will exhibit similar forwarding behavior.

To formally specify the soundness and completeness of the compression process, they model the semantics of a network's control plane based on the *stable routing problem* (SRP). An SRP instance consists of the following components:

1. a graph $G = \langle V, E\rangle$ denoting the network topology.
2. a type $A_\bot = A + \bot$ denoting the type of the routing attributes exchanged. $\bot$ means no route.
3. a destination $d : V$ and its initial announcement $a_d : A$.
4. a partial order $\prec \subseteq A_\bot \times A_\bot$ used to rank attributes.
5. a function $\texttt{transfer} : E \to A_\bot \to A_\bot$ that denotes how messages are processed across edges.

A solution to an SRP, is a labelling function $\mathcal{L} : V \to A_\bot$ which represents the best available route for each node, as captured by the constraints in fig. 1.

Label and *forwarding* equivalence (fig. 2) define the relationship between the concrete and abstract networks. While forwarding equivalence preserves qualitative properties of nodes such as reachability, path length, and absence of routing loops, it does not preserve *quantitative* properties such as the number of neighbors of a node. Hence, it cannot be directly used to reason about the network in the presence of failures.

$$\mathcal{L}(u) = \begin{cases} a_\mathrm{d} & u = d \\ \bot & \mathrm{attrs}_{\mathcal{L}}(u) = \emptyset \\ a \in min_{\prec}\mathrm{attrs}_{\mathcal{L}}(u) & \mathrm{attrs}_{\mathcal{L}}(u) \neq \emptyset \end{cases}$$

$$\begin{aligned} \mathrm{attrs}_{\mathcal{L}}(u) &= \{a \mid (e, a) \in \mathrm{choices}_{\mathcal{L}}(u)\} \\ \mathrm{choices}_{\mathcal{L}}(u) &= \{(e, a) \mid e = (u, v),\ a = \texttt{transfer}\ e\ \mathcal{L}(v),\ a \neq \bot\} \\ \mathrm{fwd}_{\mathcal{L}}(u) &= \{e \mid a \approx \mathcal{L}(u), (e, a) \in \mathrm{choices}_{\mathcal{L}}(u)\} \end{aligned}$$

where $a \approx b \triangleq a \not\prec b \wedge b \not\prec a$

Figure 1. Valid solutions of an SRP

$SRP = (G, A, a_\mathrm{d}, \prec, \texttt{transfer})$ — *concrete SRP instance*
$\widehat{SRP} = (\widehat{G}, \widehat{A}, \widehat{a_\mathrm{d}}, \widehat{\prec}, \widehat{\texttt{transfer}})$ — *abstract SRP instance*
$f : V \to \widehat{V}$ — *topology abstraction*
$h : A \to \widehat{A}$ — *route abstraction*

$\mathcal{L} \in SRP \iff \widehat{\mathcal{L}} \in \widehat{SRP}$ when:
(1) $\forall u.\ h(\mathcal{L}(u)) = \widehat{\mathcal{L}}(f(u))$ — *label-equivalence*
(2) $(\widehat{u}, \widehat{v}) \in \widehat{\mathrm{fwd}}_{\widehat{\mathcal{L}}}(\widehat{u}) \iff$ — *fwd-equivalence*
$\left(\forall u.\ f(u) = \widehat{u} \implies \exists v.\ f(v) = \widehat{v} \wedge (u, v) \in \mathrm{fwd}_{\mathcal{L}}(u)\right)$

$\mathcal{L} \in SRP <: \widehat{\mathcal{L}} \in \widehat{SRP}$ when:
(1) $\forall u.\ h(\mathcal{L}(u)) \preceq \widehat{\mathcal{L}}(f(u))$ — *label-approximate*

Figure 2. Relation between concrete and abstract SRP.

Compression in the presence of failures

One key invariant that Bonsai maintained is that all concrete nodes in an abstract node have similar forwarding behavior. In the presence of arbitrary (but bounded) failures this is no longer true, as demonstrated by the example of fig. 3. This invariant is simply too strong to be useful in the context of fault tolerance, as failures break the symmetries between nodes that Bonsai exploits to compress the network.

*This is joint work with Ryan Beckett, Ratul Mahajan and David Walker. The author of this abstract is the sole student and leader of the project.

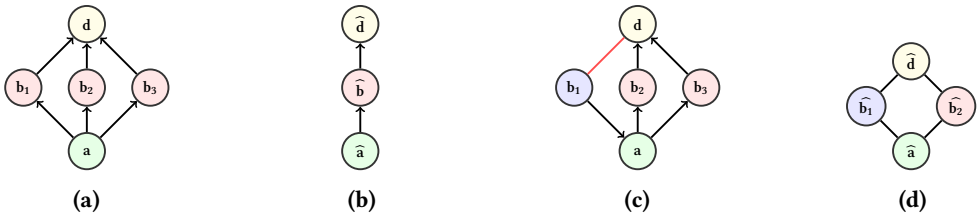

Figure 3.

(a) shows the concrete network after converging to a stable state. The direction of the arrows coincides with the direction of forwarding.
(b) shows a label and forwarding equivalent abstract network as computed by Bonsai.
(c) shows the state of (a) when the link between b_1 and d has failed. (b) and this network are no longer label/forwarding equivalent.
(d) shows a new abstraction after splitting node $\widehat{b}$. This abstraction can correctly model the reachability of nodes in the concrete network, under any possible single link failure.

Simplifying the problem. Label and forwarding equivalence no longer hold, but we make a key observation:

> If a node has a route to the destination under any k link failures then it must also have one under any k' link failures for $k' < k$.

This intuition is formalized by establishing a weaker relation between the concrete and the abstract SRPs. In particular, that the solutions of the concrete and the abstract network are *label-approximate* (see fig. 2). Intuitively any concrete node has a route that is at least as good (according to $\prec$) as the route that the corresponding abstract node has. This relation is sound when reasoning about properties such as reachability to the destination; if an abstract node can reach the destination then so can the corresponding concrete nodes.

However, this begs the question: what if an abstract node cannot reach the destination? Consider for instance the abstract network of fig. 3b when at most one link in the network can fail. If the link between nodes $\widehat{b}$ and $\widehat{d}$ fails then no abstract node can reach the destination. The fact that the solutions of the concrete (fig. 3a) and the abstract (fig. 3b) networks are label-approximate is not very useful; all concrete nodes can reach the destination but the abstract network provides no insight about this. Is label-approximate too weak to be useful?

Refining the abstraction. The original abstraction computed by Bonsai is not very useful to reason about fault tolerance properties. Notice that an abstract link represents multiple concrete links. Hence the failure of the link between $\widehat{b}$ and $\widehat{d}$ in the original abstraction (fig. 3b) corresponds to failure of the links $\langle b1, d\rangle, \langle b2, d\rangle, \langle b3, d\rangle$, a total of 3 link failures. As such it cannot constitute a real counterexample to a property that considered at most one link failure. To rectify this, we refine our original abstraction by splitting the abstract node $\widehat{b}$ into two nodes $\widehat{b_1}, \widehat{b_2}$ to obtain the abstraction of fig. 3d. This network is still a valid abstraction of the concrete network (alas a more fine-grain one); we can establish that the solutions of the two networks are label-approximate. Moreover, we can verify that any abstract node of the new abstraction (fig. 3d) can reach the destination for any single link failure and hence the same holds for any node in the concrete network (fig. 3a).

The above motivate a simple algorithm to compute an abstraction suitable for reasoning about reachability in the presence of link failures. The new algorithm is based on the idea of counterexample-guided refinement abstraction.

Evaluation and ongoing work

We found our key observation to hold for a class of networks whose routing policies are *monotonic* and *isotonic*. Intuitively monotonicity dictates that attributes can only get worse when transferred across edges (formally $a \prec \texttt{transfer}\ e\ a$). Isotonicity says that the transfer function is order preserving (formally $a \prec b \implies \texttt{transfer}\ e\ a \prec \texttt{transfer}\ e\ b$). These properties were previously used to prove convergence of routing protocols [9]. We believe that networks such as the ones deployed in modern data centers satisfy these properties.

We have currently implemented the original Bonsai abstraction, a refinement algorithm and an SMT-based verification tool based on Minesweeper [2, 3] in OCaml. Preliminary evaluation shows promise; while without compression verification for fault tolerance doesn't scale beyond a few dozens nodes and links, we can compress the network achieving a high compression ratio that allows us to efficiently verify reachability properties under one link failure for fat tree networks [1] with hundred of nodes running BGP and OSPF.

Analyzing networks with more than one link failure is challenging as they tend to require more refinement iterations because there are multiple ways to refine an abstraction. Suboptimal refinement choices can lead to a low compression rate and bad performance. We are currently exploring smarter refinement algorithms.

Other interesting avenues to explore include relaxing the monotonicity and isotonicity requirements, and studying the set of properties that can be verified via this approach.

References

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A Scalable, Commodity Data Center Network Architecture. In *SIGCOMM*.

[2] Ryan Beckett. 2017. MineSweeper Source Code. https://batfish.github.io/minesweeper. (2017).

[3] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2017. A General Approach to Network Configuration Verification. In *SIGCOMM*.

[4] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2018. Control plane compression. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 476–489.

[5] Nick Feamster and Jennifer Rexford. 2007. Network-Wide Prediction of BGP Routes. *IEEE/ACM Trans. Networking* 15, 2 (2007).

[6] Ari Fogel, Stanley Fung, Luis Pedrosa, Meg Walraed-Sullivan, Ramesh Govindan, Ratul Mahajan, and Todd Millstein. 2015. A General Approach to Network Configuration Analysis. In *NSDI*.

[7] Aaron Gember-Jacobson, Raajay Viswanathan, Aditya Akella, and Ratul Mahajan. 2016. Fast Control Plane Analysis Using an Abstract Representation. In *SIGCOMM*.

[8] Joanne Godfrey. 2016. The Summer of Network Misconfigurations. https://blog.algosec.com/2016/08/business-outages-caused-misconfigurations-headline-news-summer.html. (2016).

[9] Timothy G. Griffin and Joäo Luís Sobrinho. 2005. Metarouting. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '05)*. 1–12.

[10] Ratul Mahajan, David Wetherall, and Tom Anderson. 2002. Understanding BGP Misconfiguration. In *SIGCOMM*.

[11] B. Quoitin and S. Uhlig. 2005. Modeling the Routing of an Autonomous System with C-BGP. *Netwrk. Mag. of Global Internetwkg.* 19, 6 (November 2005), 12–19.

[12] Yevgenly Sverdlik. 2012. Microsoft: misconfigured network device led to Azure outage. http://www.datacenterdynamics.com/content-tracks/servers-storage/microsoft-misconfigured-network-device-led-to-azure-outage/68312.fullarticle. (2012).

[13] Konstantin Weitz, Doug Woos, Emina Torlak, Michael D. Ernst, Arvind Krishnamurthy, and Zachary Tatlock. 2016. Formal Semantics and Automated Verification for the Border Gateway Protocol. In *NetPL*.