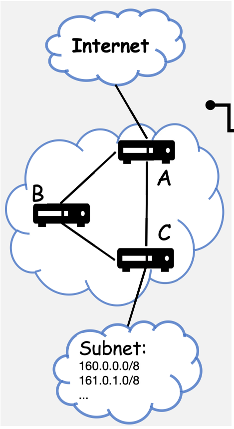
Scaling Data Plane Verification with Throughput-Optimized Atomic Predicates

Dong Guo[1] Jian Luo[2] Kai Gao[2] Y. Richard Yang[3]

[1]Tongji University [2]Sichuan University [3]Yale University

Background: Binary Decision Diagram (BDD) is a widely used data structure to represent packet header space in Data Plane Verification (DPV) systems because of its *extensibility*, *memory efficiency*, and *performance*.



Data Plane Verification

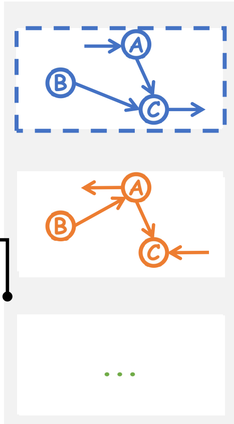
Collecting data plane configuration
(forwarding table, ACL, etc.)

↓

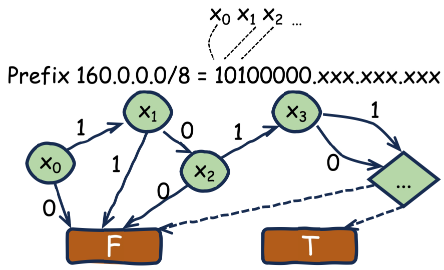
Constructing forwarding graph for each packet equivalent class

↓

Verifying properties based on the forwarding graphs
(loops, black holes, path regex, etc.)



BDD is an efficient data structure to represent and manipulate *header spaces of equivalent classes* (aka *atomic predicates*). It encodes a header space as a directed acyclic graph of variables, and allows efficient union, intersection and negation of header spaces.



Motivation & Key Insight: BDD performance is critical to DPV performance but existing BDD libraries are *not designed* to fully leverage parallelism to scale up DPV.

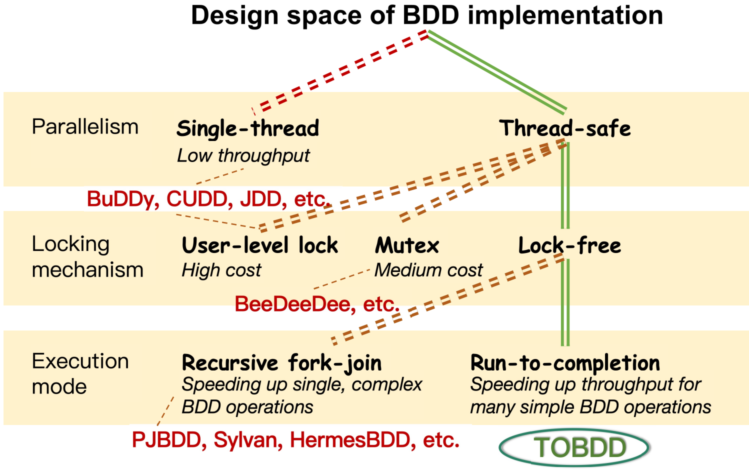
DPV's scalability requirements on BDD libraries:

1. **Bursts of BDD operations:** Many BDD operations may be triggered by a single update.
2. **Atomic semantics:** A single update is completed only when all BDD operations are executed.
3. **Simple operations:** BDD operations in DPV are mostly simple operations, *i.e.*, union, intersection, negation.

Key Insight:

BDD libraries for DPV must be designed to optimize ***throughput*** (*i.e.*, BDD operations per unit time)

Design space of BDD implementation



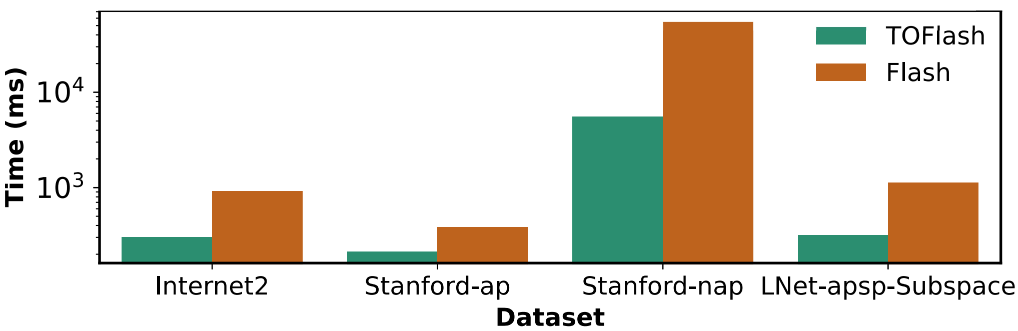
Implementation & Evaluation: TOBDD effectively speeds up state-of-the-art DPV with parallelism, improving scalability.

TOBDD implementation highlights:

- **Reschedulable Thread-pool:** Eliminating the overhead of frequent thread creation and destruction
- **Lock-free Node Table and Cache:** leveraging Compare-and-Swap (CAS) instruction to achieve lock-free BDD operations
- **Run-to-Completion Mode:** enabling operator-level rather than subtree-level parallelism for BDD oprations
- **Commutative Hash Function:** reducing cache misses

Evaluation:

Participants: We compare our previously implemented Flash artifact (JDD with user lock) with TOFlash (TOBDD) on various datasets covering WAN (Internet2), Campus (Stanford-*), and data center (LNet-apsp-Subspace).
Metric: Cold-start model construction time.
Conclusion: TOBDD speeds up Flash by 2-10x on a 20-cores server.



Code, docs and issues at
https://github.com/guodong/tobdd
Group mailing list at
tobdd@googlegroups.com

Full-text available
https://bit.ly/TOBDD