(Ir)relevance reasoning for software-defined network

Anduo Wang*   Jason Croft†   Xueyuan Mei†
Matthew Caesar†   Brighten Godfrey†

*Temple University       †University of Illinois Urbana-Champaign
software-defined networking (SDN)
software-defined networking (SDN)

SDN moves complexity to control software: an opportunity and challenge.
software-defined networking (SDN)
SDN control software

- monitor
- violation
- computation
- updates
- reconfigure
- network
SDN control software

an individual control operation

violation

updates
SDN control software

violation → updates
violation → updates
violation → updates
violation → updates

grouped into module
managing complexity in control software

dependency occur within and across modules
- modular programming abstraction [NSDI’13, 15; SIGCOMM’14,15]
- limitation: manual, requires understanding of module internals
managing complexity in control software

multiple dependencies can conflict
- conflict resolution: module-level priority [many popular control platforms]
- limitation: coarse-grained, manual
managing complexity in control software

updates can go wrong
- debugging and verification [SIGCOMM’14, NSDI’13, 15, 16]
- limitation: identify incorrect network events/states but not revealing incorrect control logic, post-mortem
automated reasoning support

- **automated**: reduce human involvement with formal tool (SMT solver)
- **finer-grained**: operation-level
- **static**: prior-to deployment,
- **logic based**: derive proper interactions among controls
example reasoning task: dependency

operation A depends on B
(1) A update can activate B
(2) B update never activates A
operation A depends on B

(1) A is relevant to B: can find a B update such that violates A

(2) B is irrelevant to A: cannot find a B update that violates A
formal model

SDN control loop

- network
  - monitor
    - violation
      - computation
        - updates
          - reconfigure

- view maintenance
  - view
    - database base (stored) tables
      - trigger
        - insert / delete
          - view update

a unified database representation

*ravel: a database-defined network* [SOSR’16]

[ravel-net.org](http://ravel-net.org)
dependency and irrelevance reasoning

dependency in SDN

irrelevant database updates
(ir)relevant database update

**view**

```
SELECT
FROM tables
WHERE \( C_v \)
```

**(ir)relevant update if**

\( C_v[t] \) is (UN)SAT

**update**

```
INSERT INTO table \( t \)
```

**view**

```
SELECT
FROM tables
WHERE \( C_v \)
```

**(ir)relevant update if**

\( C_v \land C_d \) is (UN)SAT

**update**

```
DELETE FROM table
WHERE \( C_d \)
```
usage: synthesize orchestrator

depends on

module 1

module 2

module 3

construct dependency graph

topological sort

- remove conflicts with user guidance
- assign each update a stratum number

synthesize a master orchestrator

- activate an update only when all updates with smaller stratum numbers have completed
usage: reasoning with partial information

conflict-free guarantee
if \( \neg C_A \supset \neg C_B \), A is guaranteed to be irrelevant to B
(corollary: synthesize conflict-free updates for A regarding B by rewriting \( C_A \) to \( C_A \lor C_B \))

feasibility of conflict-free update
if \( \neg C_A \land \neg C_B \) is SAT, there exists some A update that is irrelevant to B

infeasibility of conflict-free updates
if \( \neg C_A \land \neg C_B \) is UNSAT, no A update exists that is irrelevant to B
thank you
backup
open questions

obtain the database representation

- use *Ravel*, a database-defined control platform
  - [ravel-net.org](http://ravel-net.org)

extract the database representation from arbitrary control software

- manually construct a map between data objects and database tables
- automatically convert data updates to DB write with conditions?
- extract view condition?
limitation

distribution and concurrency

- the network data plane is a distributed system with concurrent updates
- SDN relies on multiple controller for scalability

combine DB concurrency control and irrelevance reasoning?