OF.CPP: Consistent Packet Processing

Peter Perešíni @ EPFL

Maciej Kuźniar, Nedeljko Vasić @ EPFL
Marco Canini @ TU Berlin / T-Labs
Dejan Kostić @ IMDEA Networks
Updating Network State

Transient forwarding inconsistencies

- loops
- blackholes
- security violations
Network Consistency Work

Abstractions for Network Update
(Reitblatt et al., SIGCOMM'12)
- introduces Per-packet Consistency

Incremental Consistent Updates
(HotSDN'13)

A Safe, Efficient Update Protocol for OpenFlow Networks
(McGear, HotSDN'12)
Per-Packet Consistency
What is the "Configuration"?

Network Configuration =

Switch rules/flowtables + Controller state

Previous work + This work
Our Focus = Reactive Controllers

Reactive & Centralized
● install paths by observing Packet-In events

Future work
● proactive controllers
● distributed controllers
● both consistent updates & controller consistency at the same time
POX Example: ARP Multiplication

\[ H_2 = \text{right} \]

CTRL

unexpected rule is installed

&&

packet copy is sent to H2
ARP Multiplication Storm

Copies == Performance Problems
(switch rules & controller load)
Is it Only Performance?

Rules not on the shortest path!

Unexpected behavior
Root Cause

Packet processing == multiple events
● programmer assumes non-interleaved execution
● but ctrl state can be changed by concurrent events

Two different configurations!
If we had Isolation...

Packet 1:
H1 = left
H2 = ???

Packet 2:
H1 = left
H2 = right
Isolation + Consistency?

Does it sound familiar?

- Databases & Transactions!
  - ACID properties
Challenge 1: What Belongs to a Transaction?

Transactions should be implicit
• i.e. no need to manually identify events

Our approach
• Transaction == All events caused by the same packet / its copies
Challenge 2: Can we have Atomicity?
Can we have Atomicity?

Packets are part of the transaction

- atomicity -> packet buffering

=> We need to relax atomicity!
Challenge 3: Relaxed Atomicity & Rollback

- full rollback is impossible
  - but do we need it?

- single event rollback
  - possible
  - no need to expensively buffer packets
  - rollback is perceived by the controller as packet loss
Challenge 4: Relaxed Atomicity & Isolation

Non-atomic transactions

- send egress packets before the transaction finishes
- problem = causality
  - packets can cause a new transaction
  - should it we process it in isolation?
Problem: Isolation vs. Causality

Initial state: H1 = ???, H2 = ???

T1:
H1 = left
H2 = ???

T2:
T2 needs to see state of T1
Putting it together:

Use Transactions
- automatically infer related events

Multi-Commit
- commit / abort after each event

Preserve causality
- see state from older transactions (even if they did not finish)
OF.CPP Prototype

- in Python
- identifies transactions using VLAN tags

- applied to POX controller
  - `l2_multi` and `arp_responder` modules
  - solves ARP multiplication problem
Summary

Consistency is important
● hard to get it right

● our approach: Multi-commit Transactions
  ○ works well for reactive controllers

● plenty of space for future work!
Backup slides
Proactive?

Similar problems as reactive

- controller action may span several events
ARP multiplication - It happens in reality
Inconsistency causing loop
Per-packet consistency

During any network change a packet should be processed according to

- **old** configuration; or
- **new** configuration;
- but **never a mixture** of the two
Transactions - serialization order