SERGEANT

A transactional distributed kernel for software defined networks

Overview

Vision

The controller as an OS that safely runs many network applications from many sources (e.g., partially trusted, non-experts)

Model

Programmer sees single-threaded view

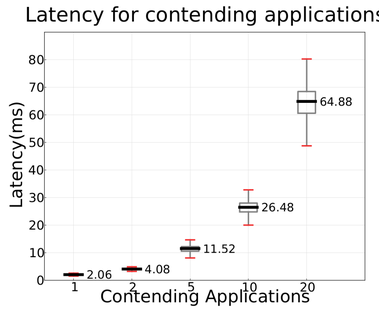
Updating hardware = writes to software objects

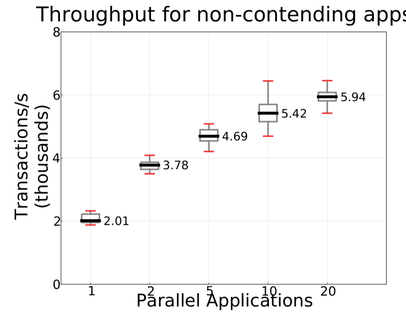
RPCs to other controllers included in transaction (nested, distributed transactions)

Evaluation

Agreement, consistency, isolation, fairness with adequate performance

Single Controller

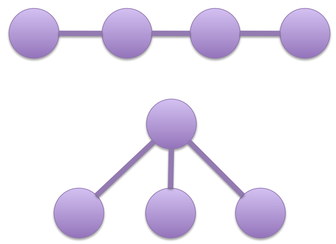




Up to 6k transactions/s on a controller.

Hundreds of transactions/s per switch.

Multiple Controllers



~6.82 ms/program

~4.94 ms/program

Time difference from parallelization of two-phase commit. (Ask about this.)

All experiments in mininet; ~.5ms RTT for multiple controllers

Four Features of SERGEANT

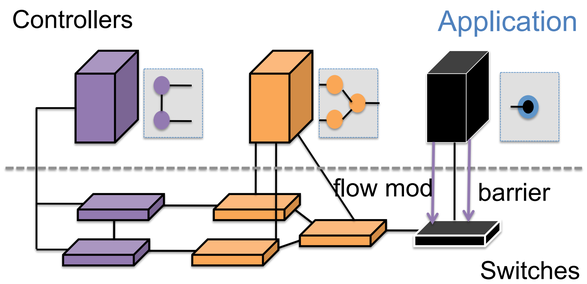
Agreement

between hardware state & software representation of it

Today: Programmer

- Issue barriers
- Handle errors

Sergeant: System-enforced



Automatic transactions to hardware using barriers for dirty state.

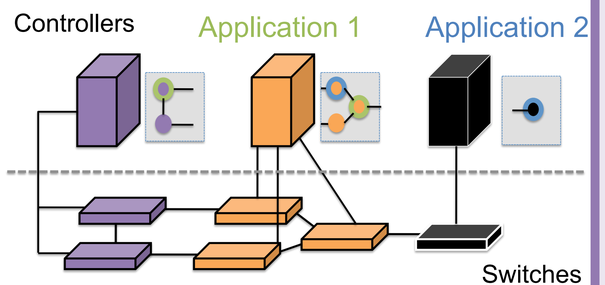
Consistency

of software state as many apps modify it

Today: Programmer

- Locks
- Database

Sergeant: System-enforced



Fine-grained locks on state automatically prevent read/ write conflicts.

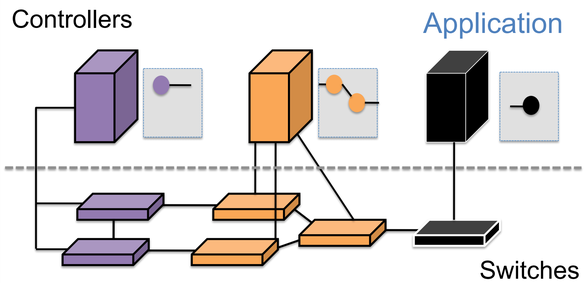
Isolation

between different applications

Today: Programmer

- ???

Sergeant: System-enforced



Permissions hide switches from apps not allowed to operate on them.

Fairness

guaranteed between applications accessing underlying resources

Today: Programmer

- ???

Sergeant: System-enforced

Novel transaction scheduling algorithm provides fairness.

Behram Mistree, Daniel Jackoway, & Philip Levis