Deploying OpenFlow @Google

Subhasree Mandal

October 17, 2011

Outline

Google

- What we did
- Challeges on the way
- Case study
  - Integration with legacy protocols
  - Time sensitivity
  - Fault tolerance: OFC failover
- Questions?

What We Did

Google

- Built an experimental network
  - WAN functionality with OpenFlow
  - Proactive flow programming

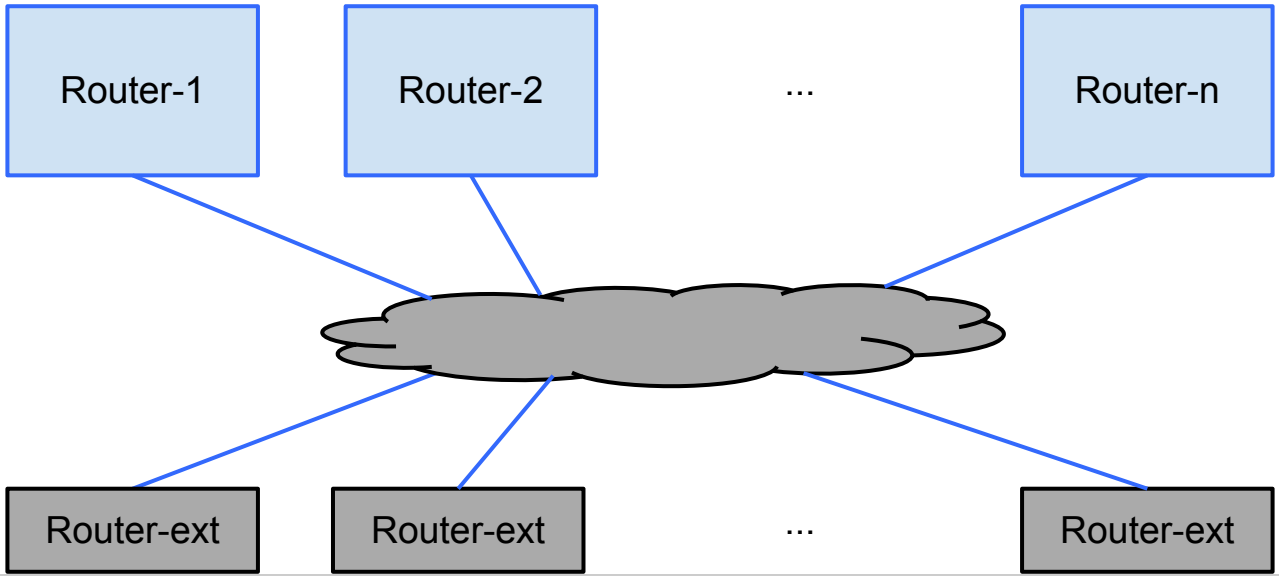
- Why
  - Utilize more powerful compute resources
  - Supports easier upgrade, experimental features
  - Target loop-free, sequenced, centralized routing
  - Enables advanced features like centralized TE

Challenges

Google™

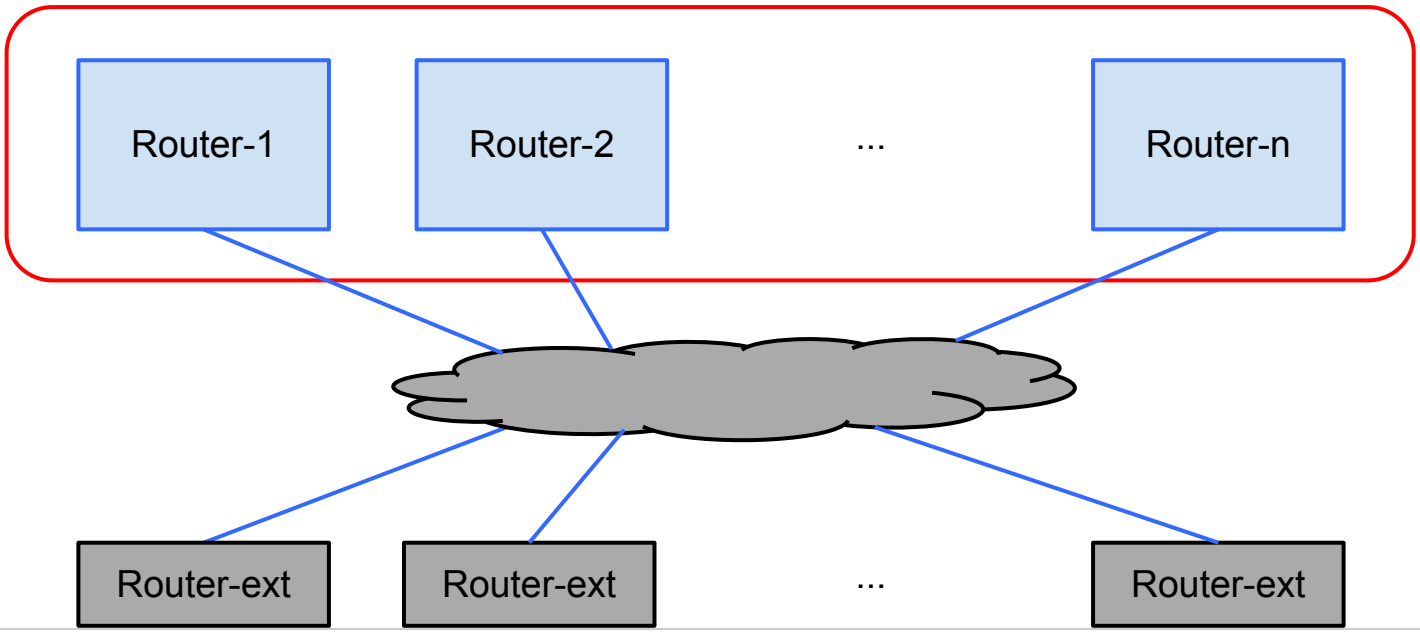
- Lagacy system:
  - Protocol speaking neighbors
  - Gradual phase out
- Speed mismatch between fast controller and slow embedded devices
- Fault tolerence: distributed vs centralized paradigm

Integration with Legacy

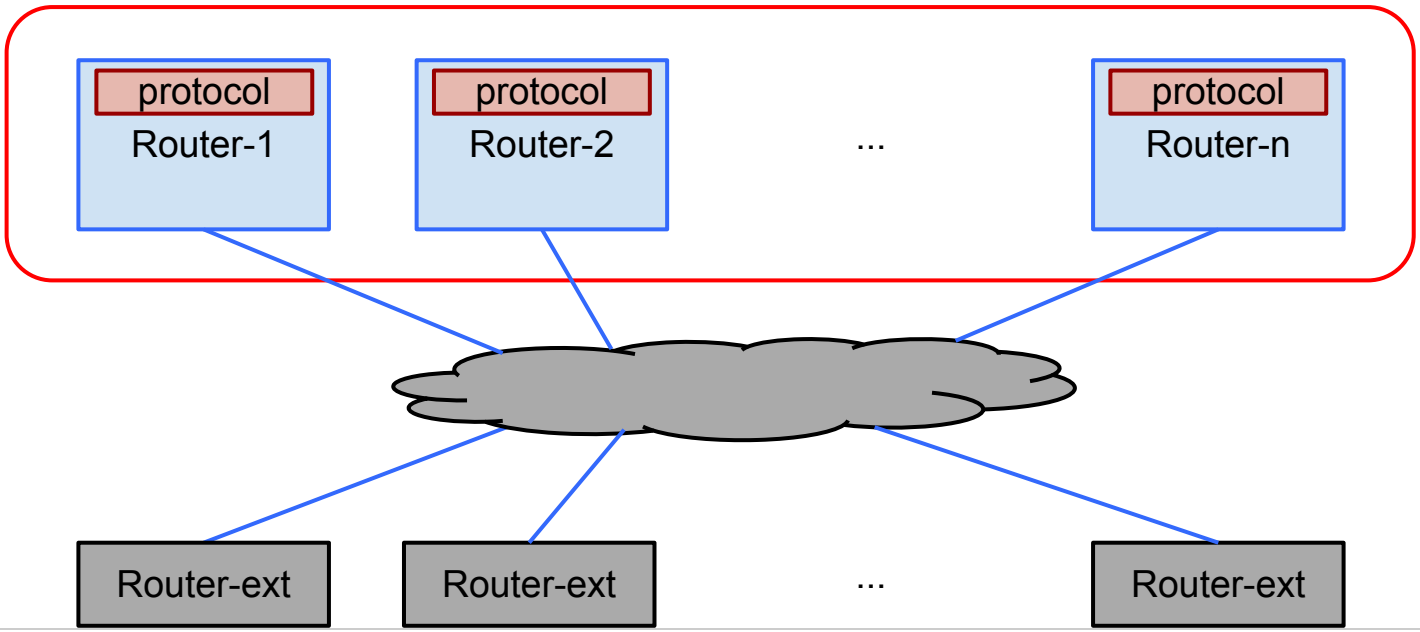


Integration with Legacy

Google™

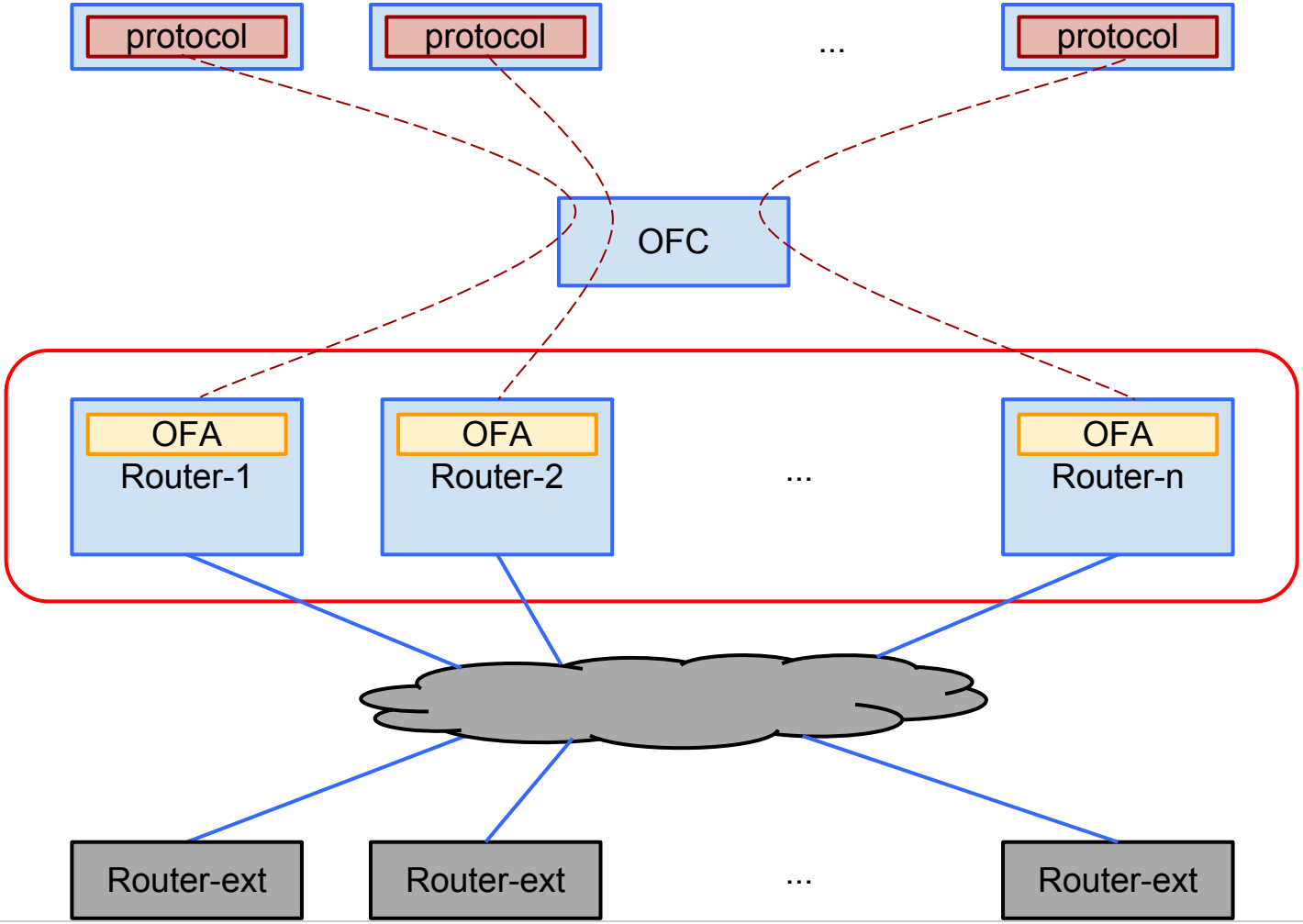


Integration with Legacy



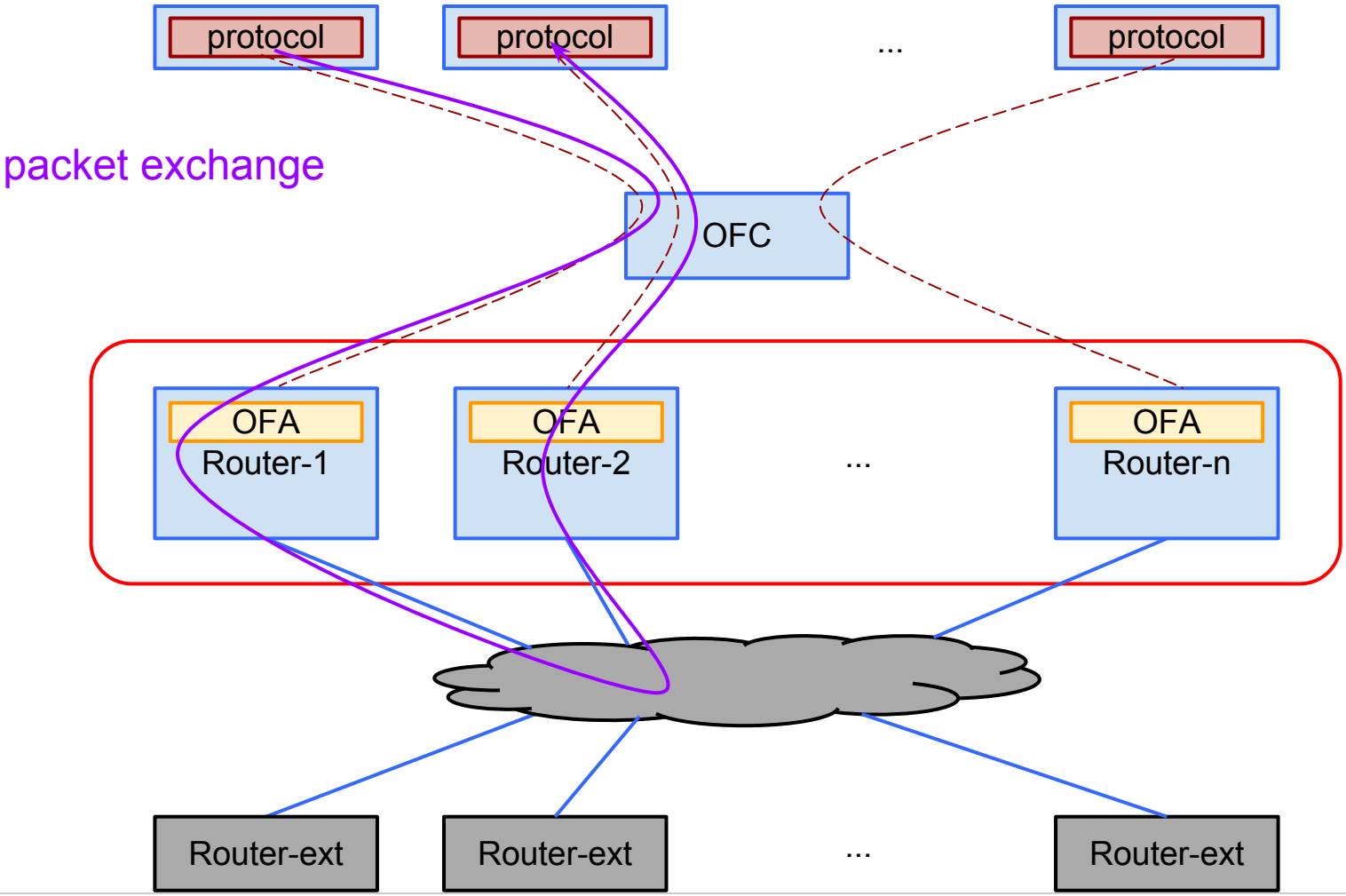
Integration with Legacy

Google™



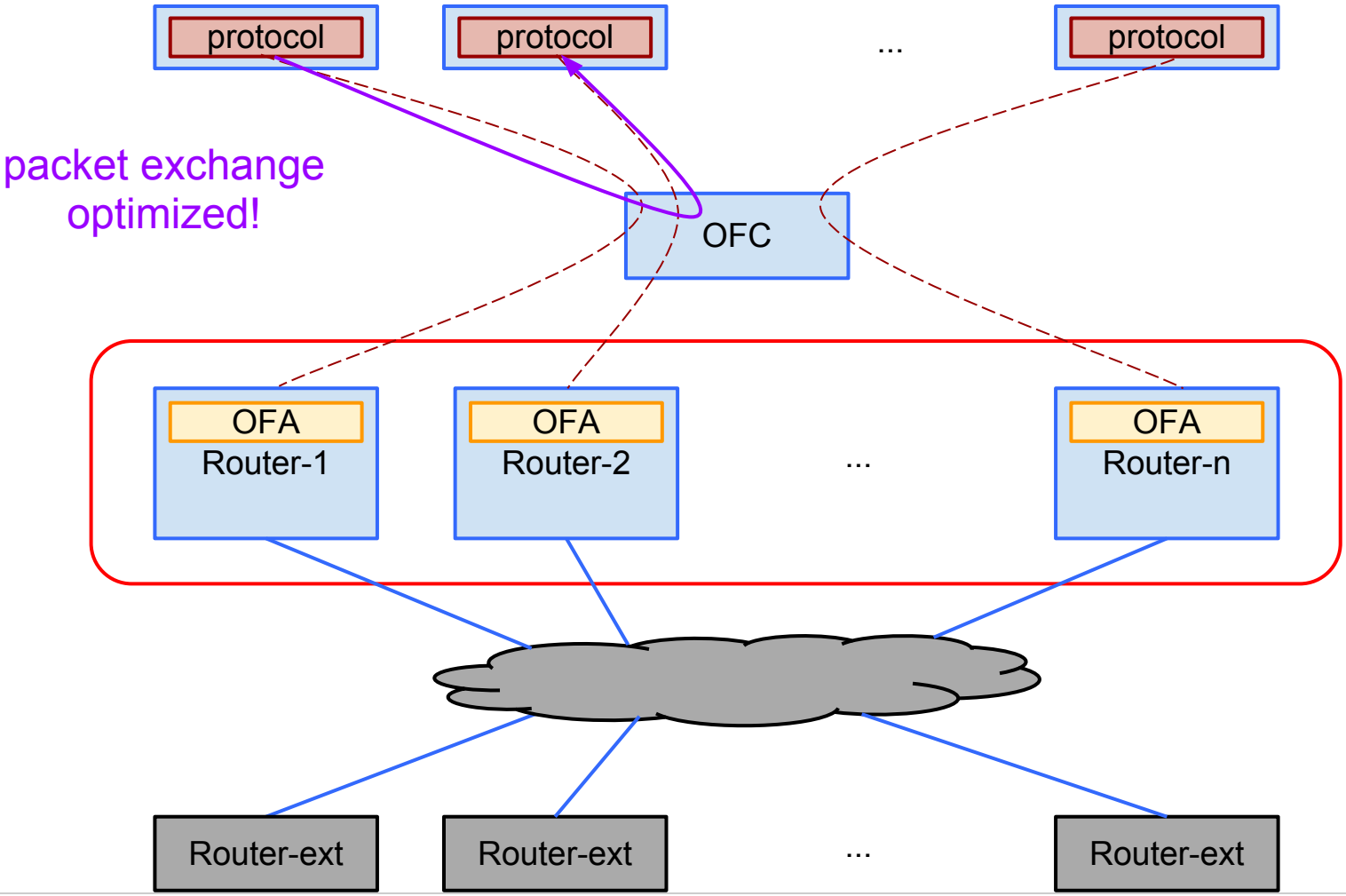
Integration with Legacy

Google™



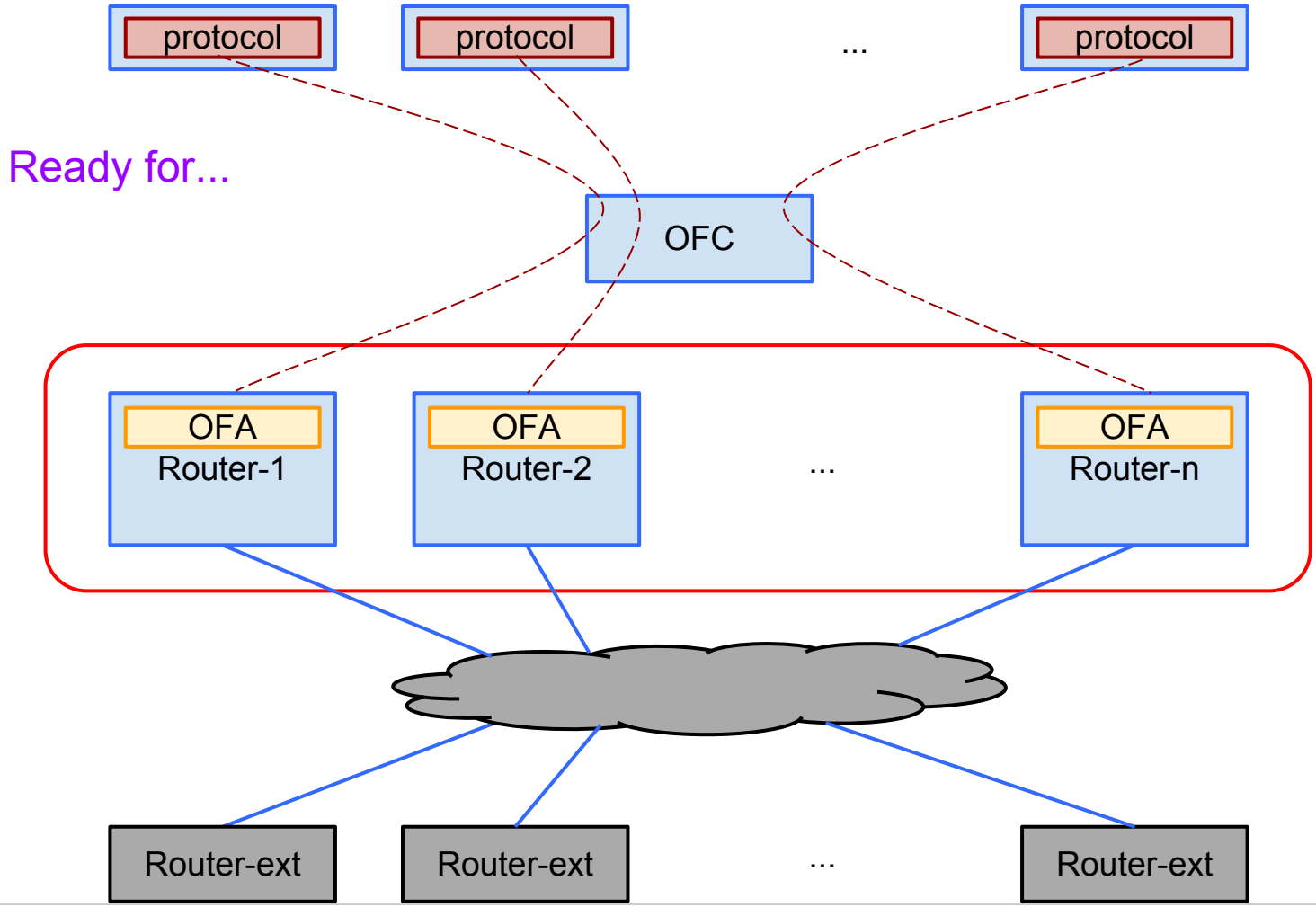
Integration with Legacy

Google™



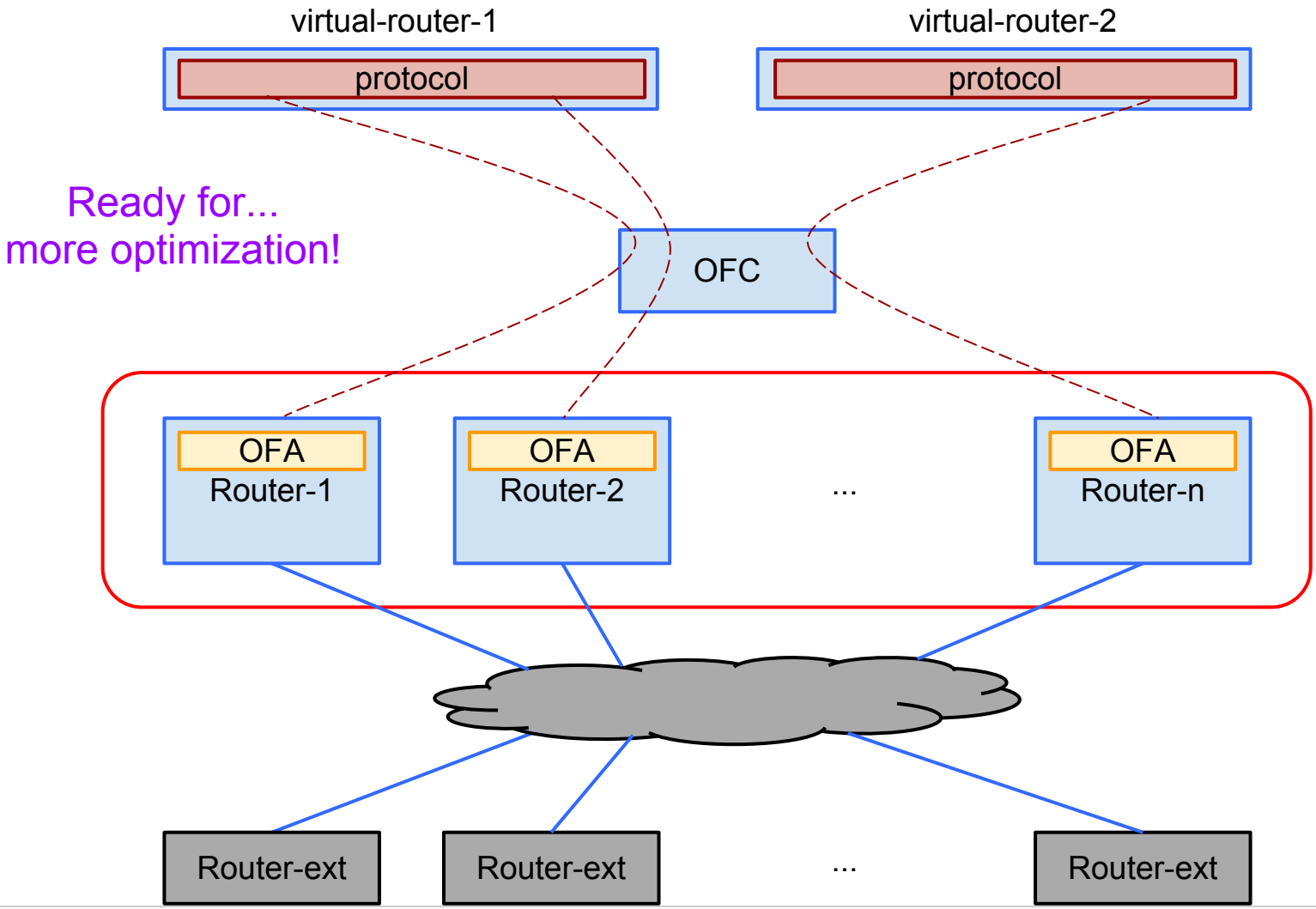
Integration with Legacy

Google™



Integration with Legacy

Google™



Integration with Legacy

Google

- Start with legacy routing protocol
- Retain backward-compatibility with neighbors
- Move to semicentralized approach: piecewise upgrade
- Gradually consolidate protocol stacks
  - Centralize intra-domain routing
  - Protocol spoken only to external neighbors

Fewer virtual routers to manage eventually

Time Sensitivity

Google™

controller

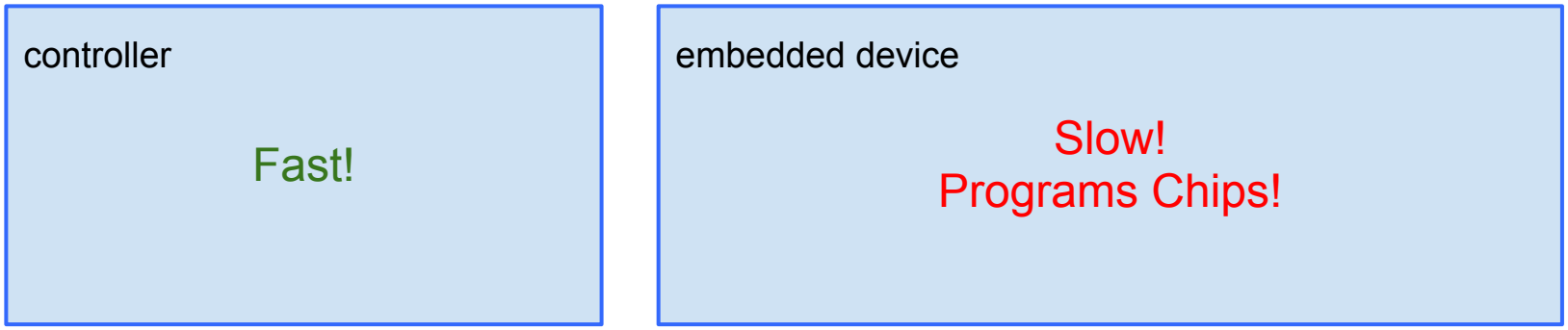
Fast!

embedded device

Slow!
Programs Chips!

Time Sensitivity

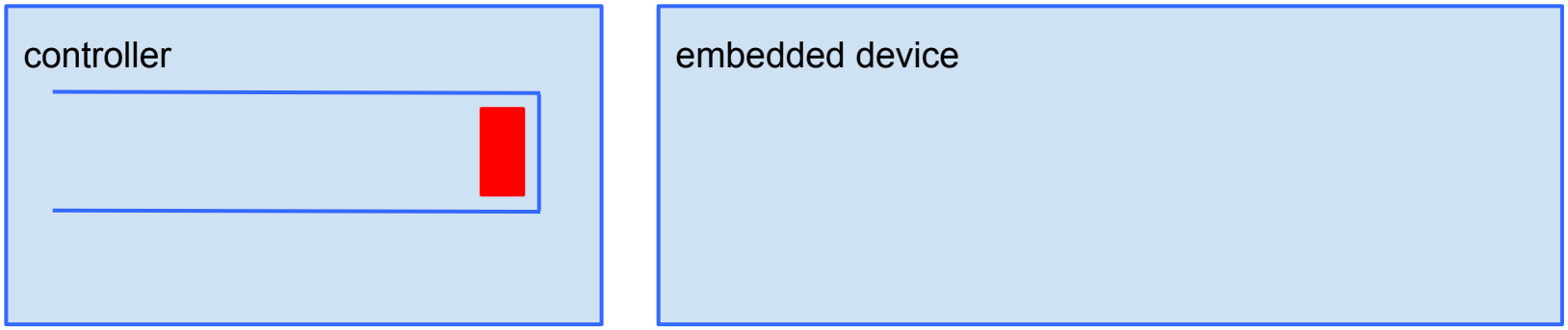
Google™



packet-out message

flow-mod message

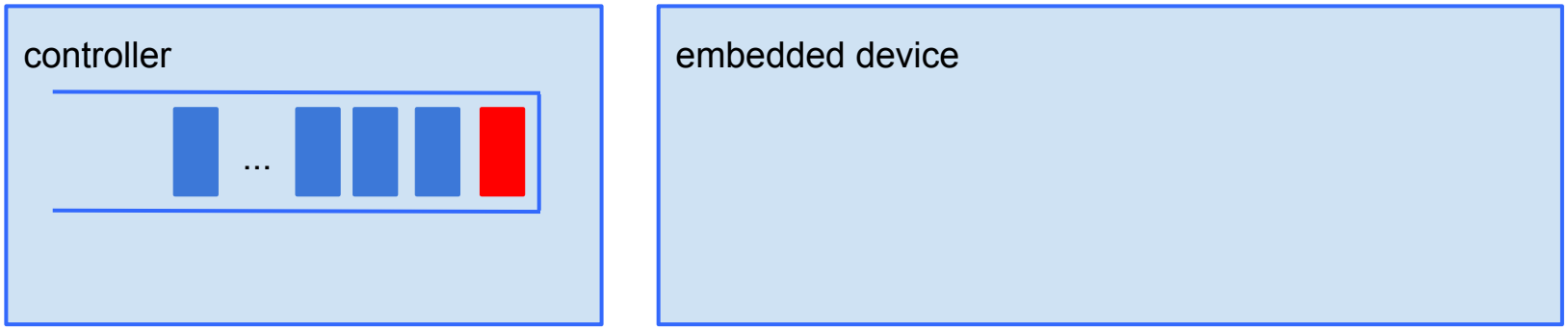
Time Sensitivity



packet-out message

flow-mod message

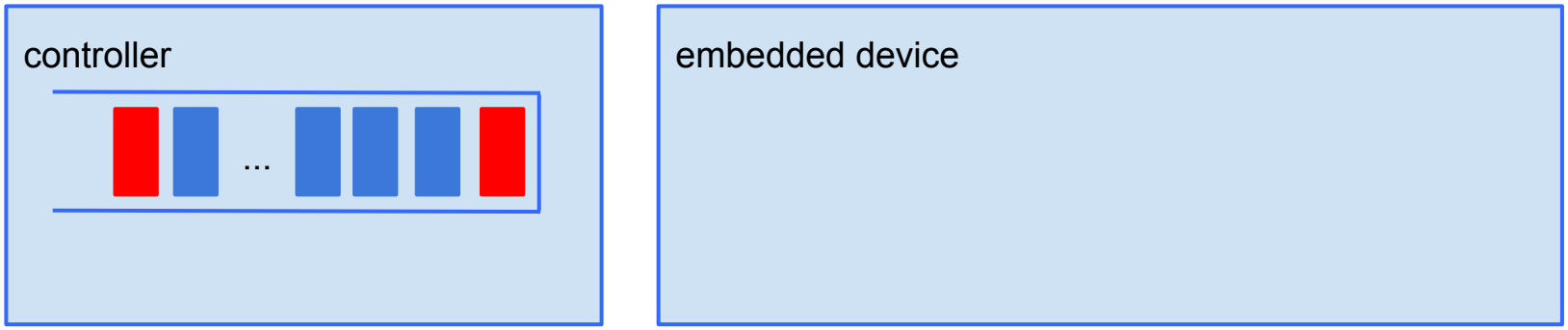
Time Sensitivity



packet-out message

flow-mod message

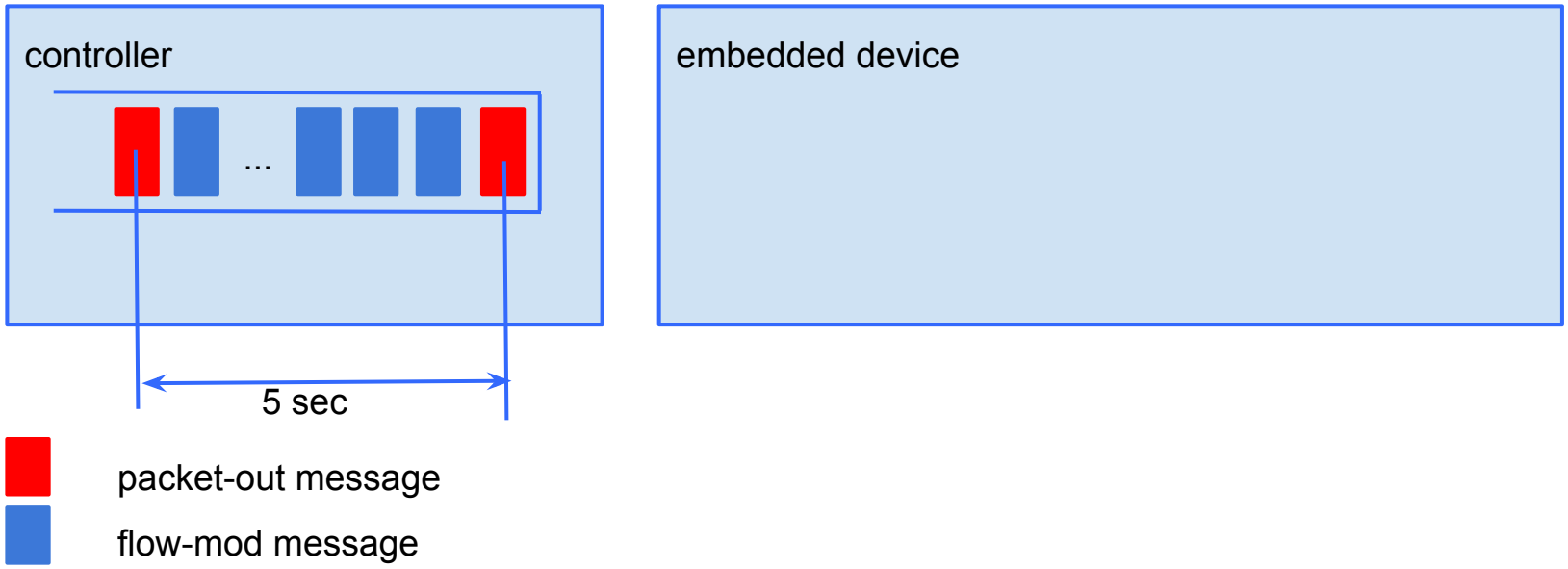
Time Sensitivity



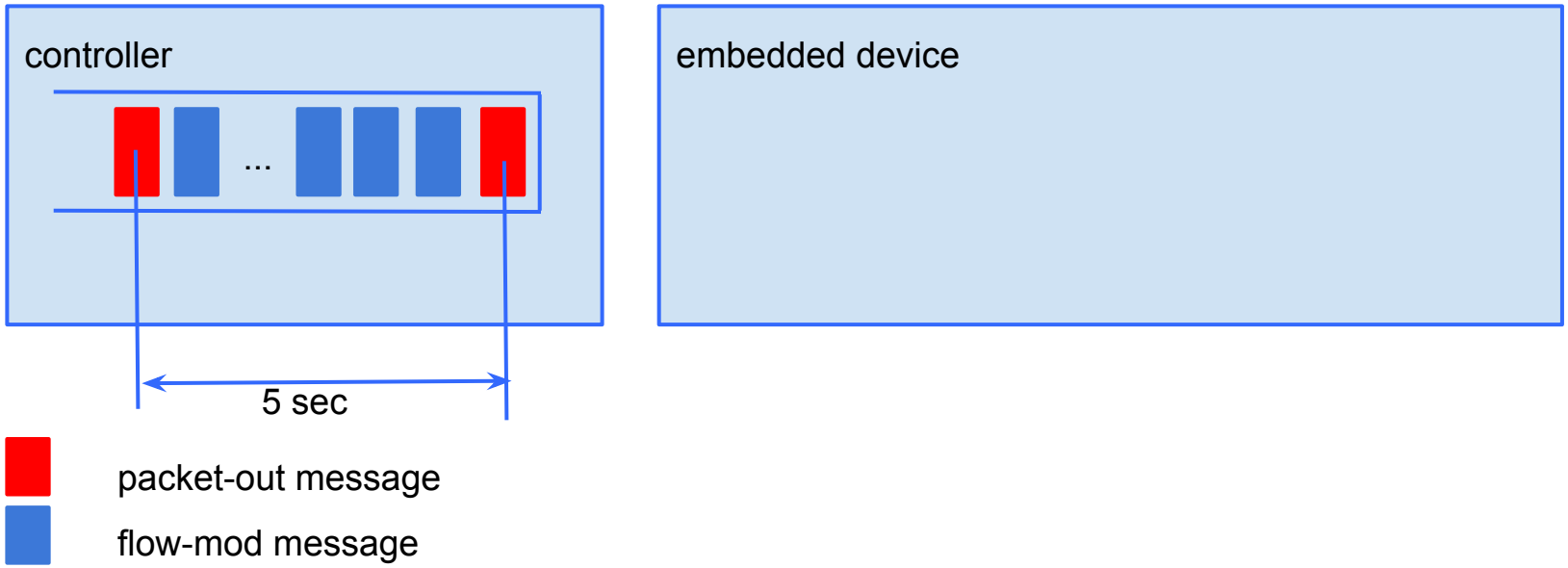
packet-out message

flow-mod message

Time Sensitivity

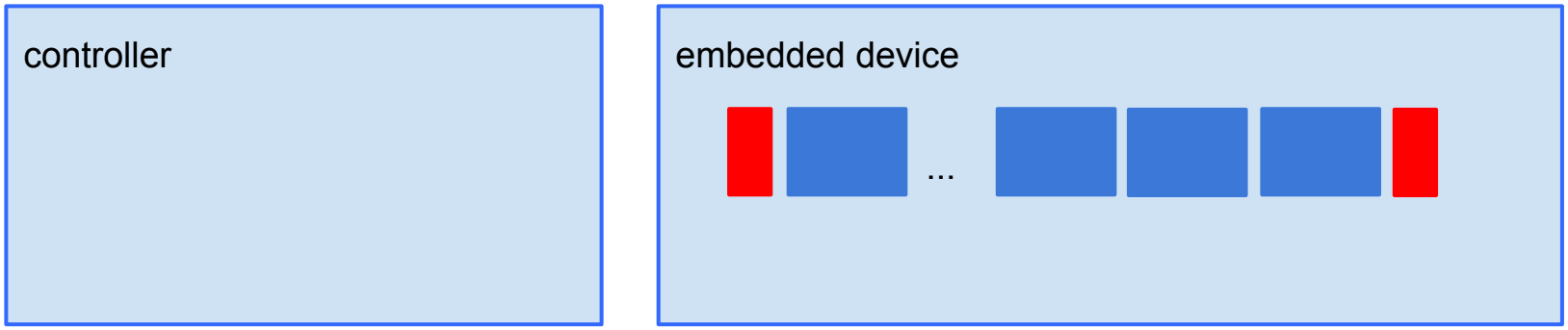


Time Sensitivity



- Fast controller enqueuing packets behind flow-mods

Time Sensitivity

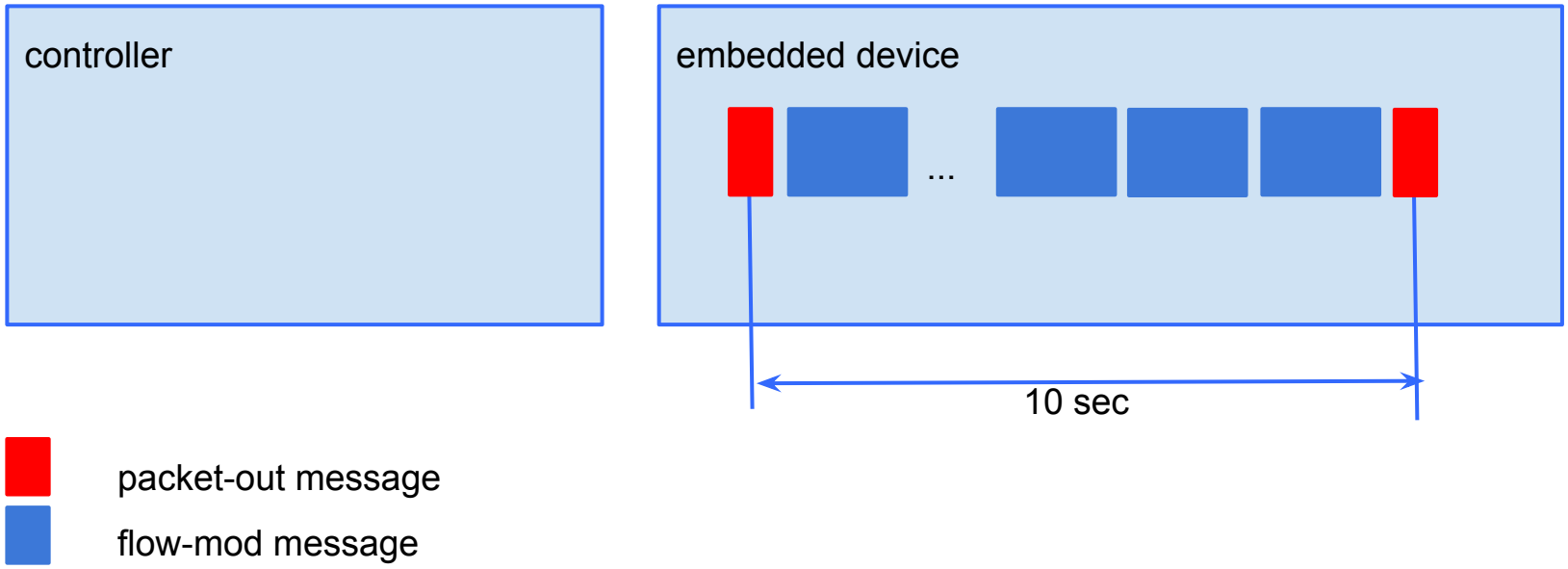


packet-out message

flow-mod message

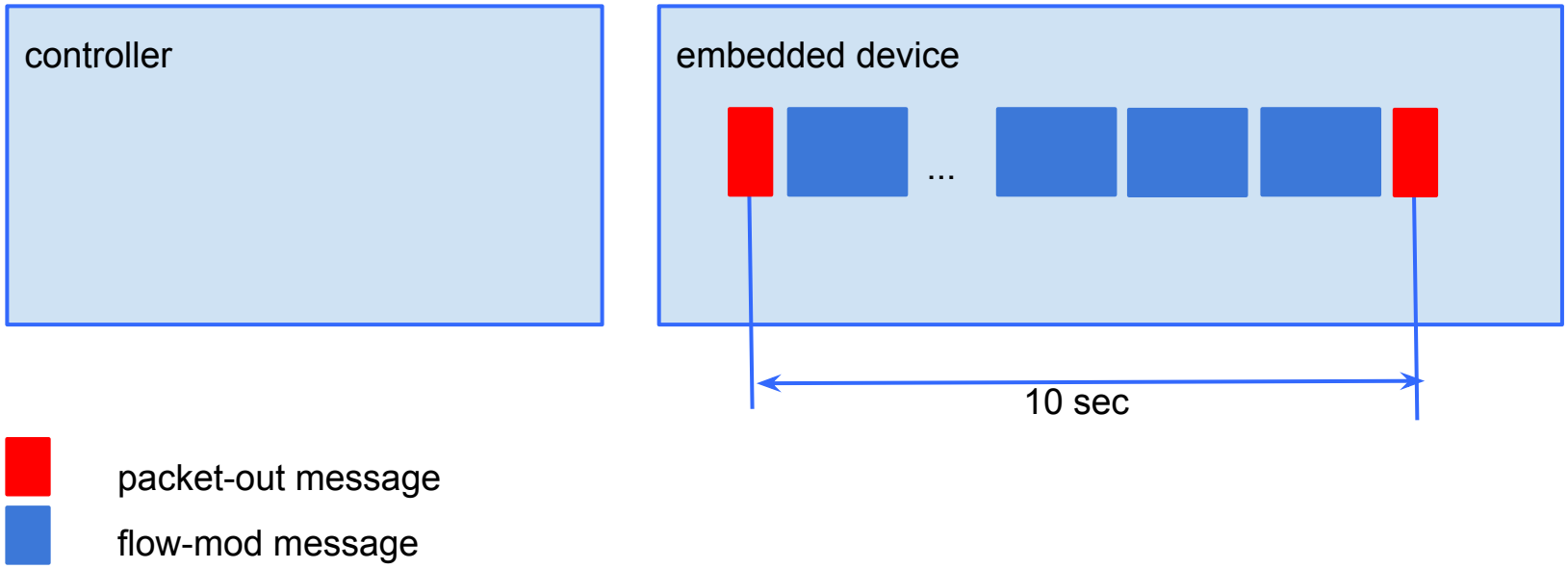
- Fast controller enqueuing packets behind flow-mods

Time Sensitivity



- Fast controller enqueuing packets behind flow-mods

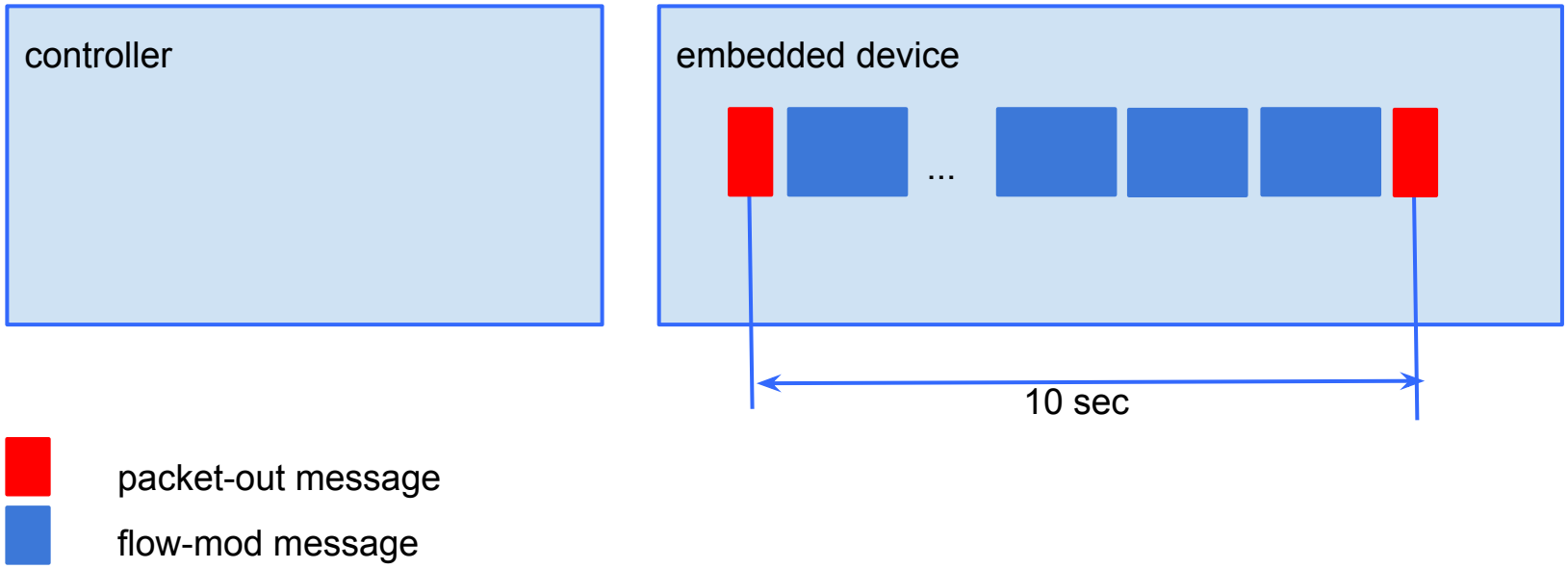
Time Sensitivity



- Fast controller enqueuing packets behind flow-mods
- Slow embedded processor programming hardware

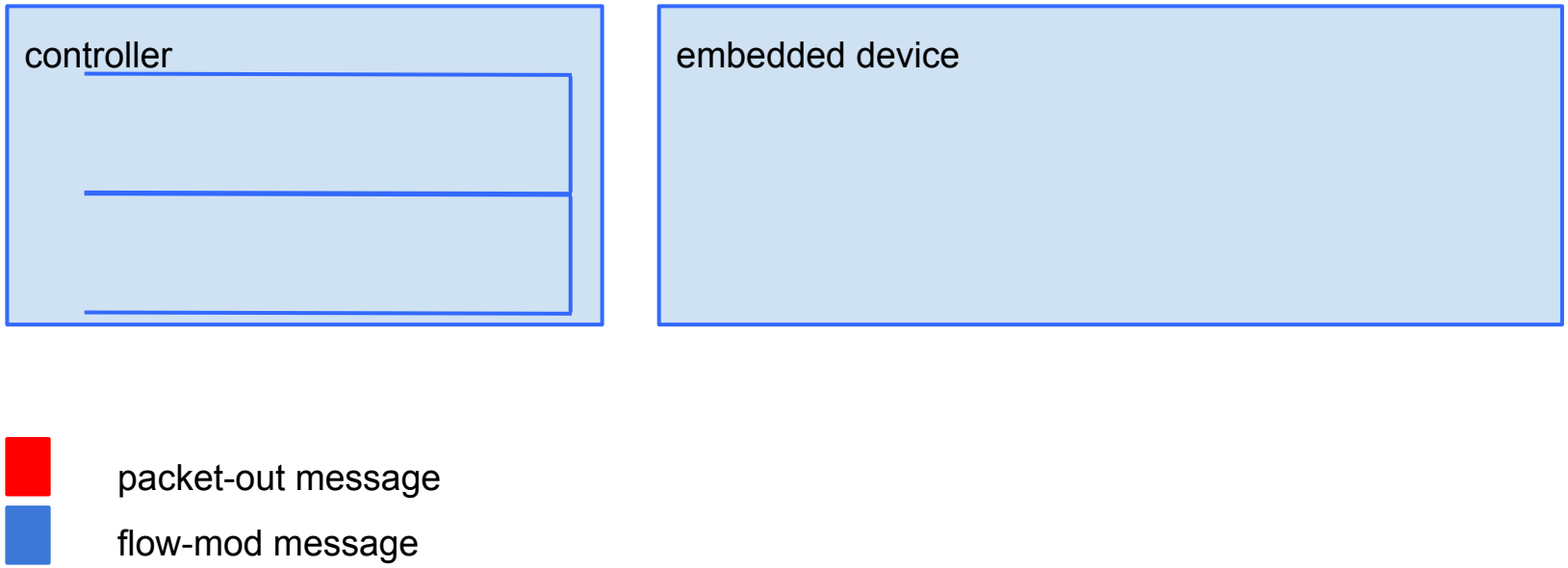
Time Sensitivity

Google



- Fast controller enqueuing packets behind flow-mods
- Slow embedded processor programming hardaware
- Time sensitive protocol packets may get delayed

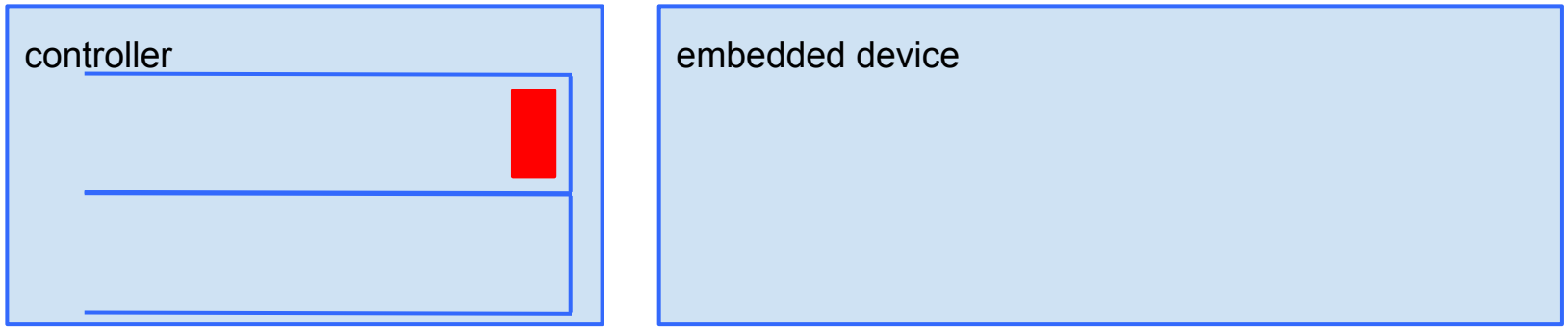
Time Sensitivity



- Maintain separate queues

Time Sensitivity

Google™



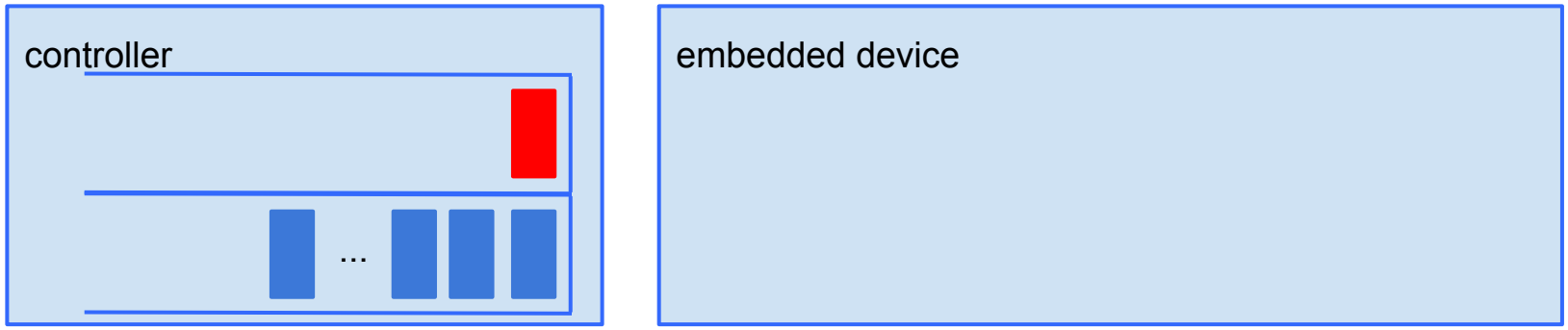
packet-out message

flow-mod message

- Maintain separate queues

Time Sensitivity

Google

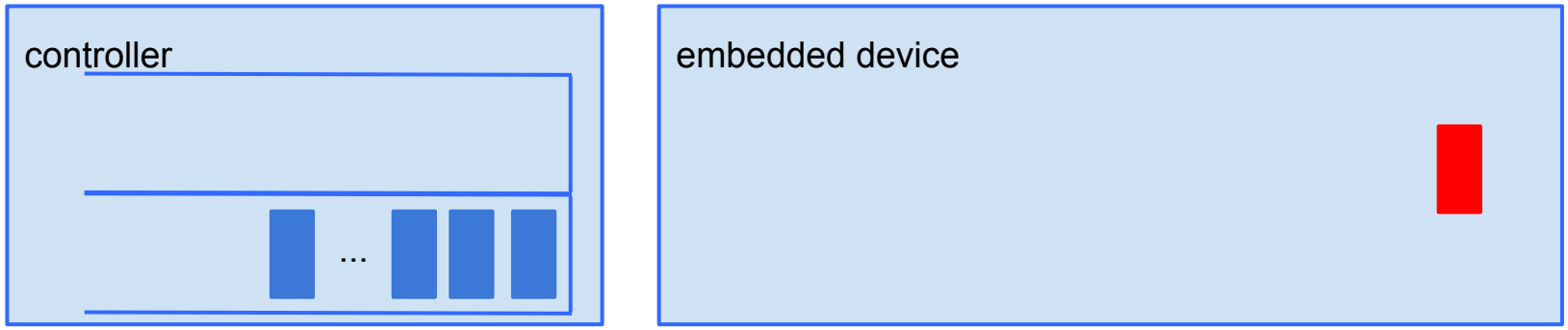


packet-out message

flow-mod message

- Maintain separate queues

Time Sensitivity

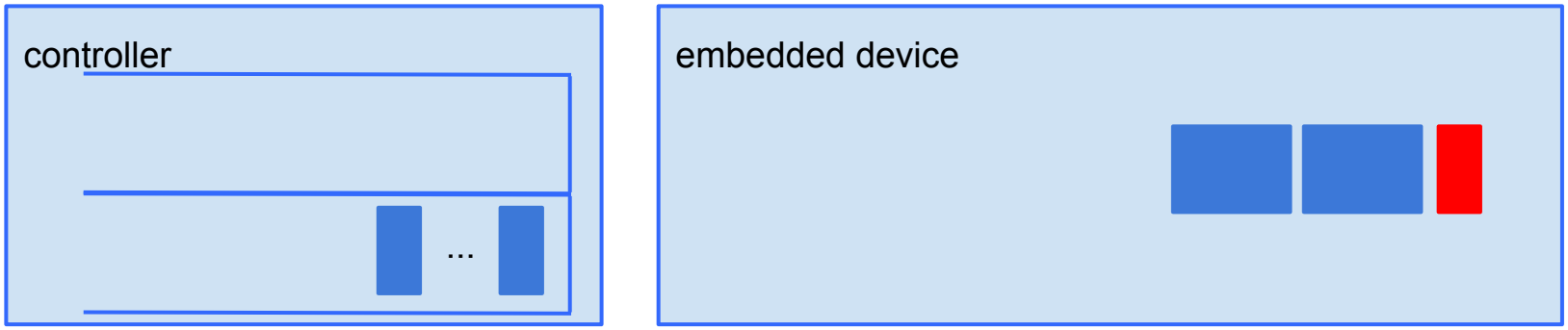


packet-out message

flow-mod message

- Maintain separate queues

Time Sensitivity

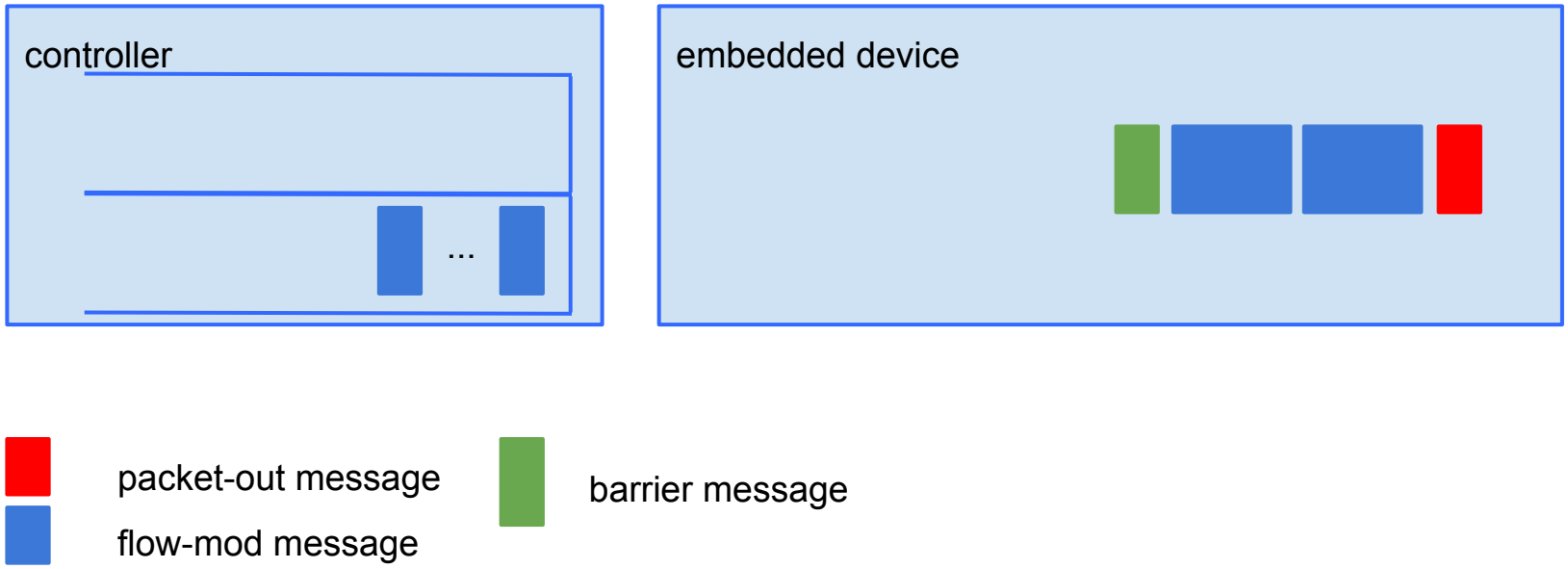


packet-out message

flow-mod message

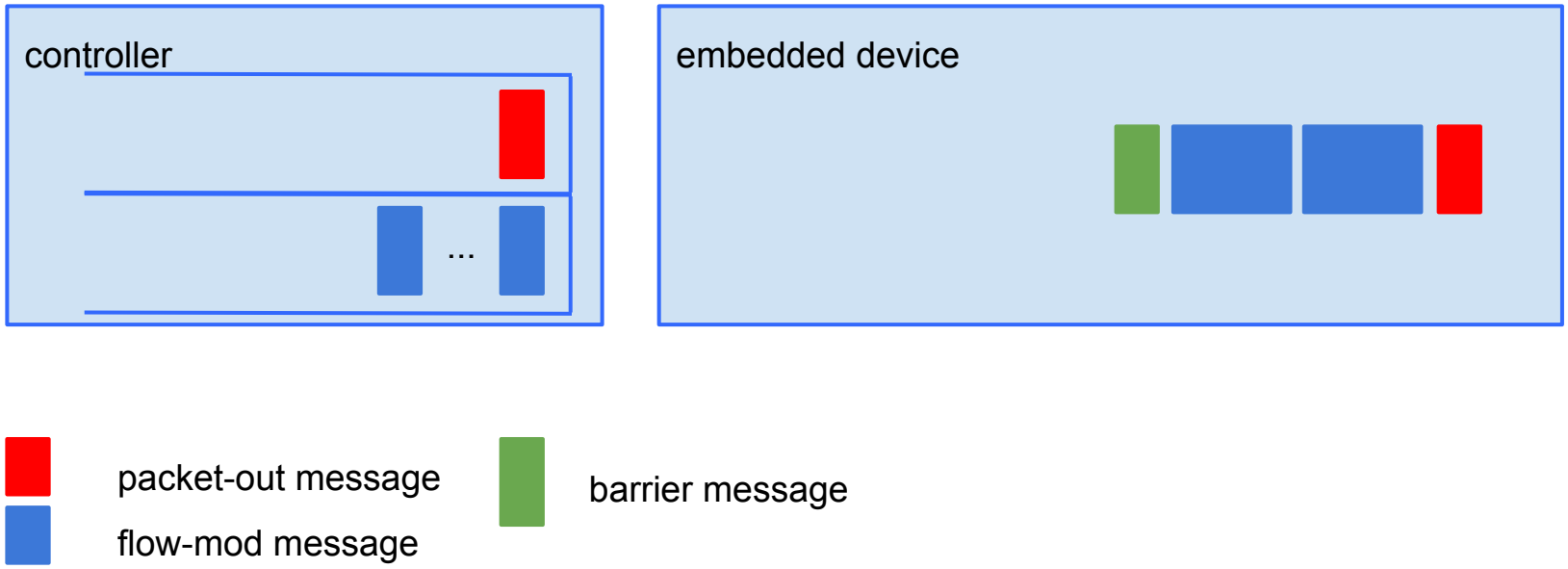
- Maintain separate queues

Time Sensitivity



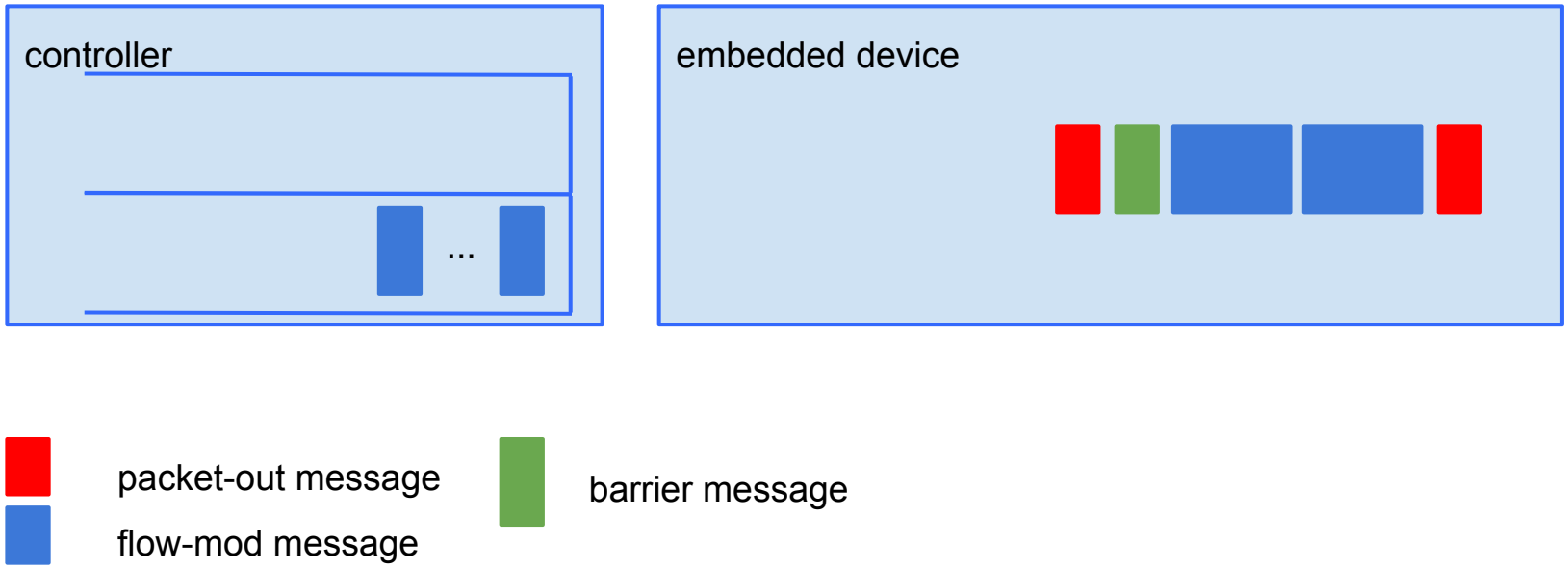
- Maintain separate queues
- Flow control using barriers: limit #outstanding flow-mod

Time Sensitivity



- Maintain separate queues
- Flow control using barriers: limit #outstanding flow-mod

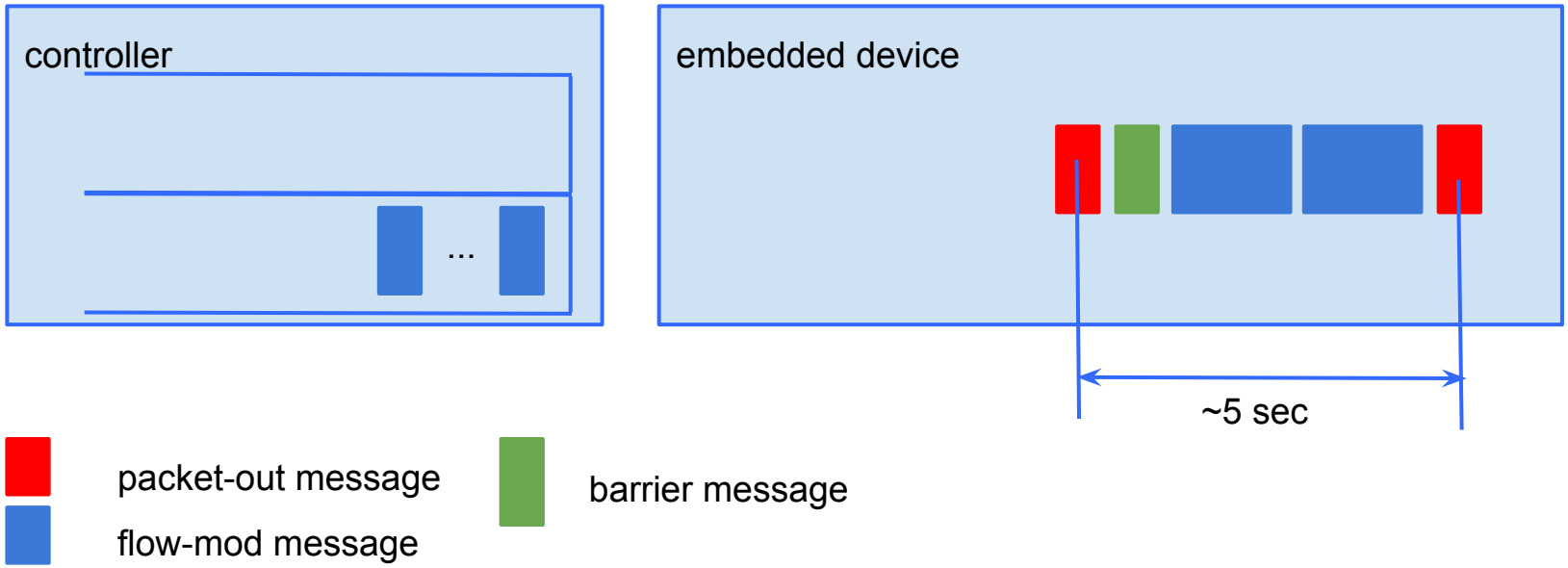
Time Sensitivity



- Maintain separate queues
- Flow control using barriers: limit #outstanding flow-mod

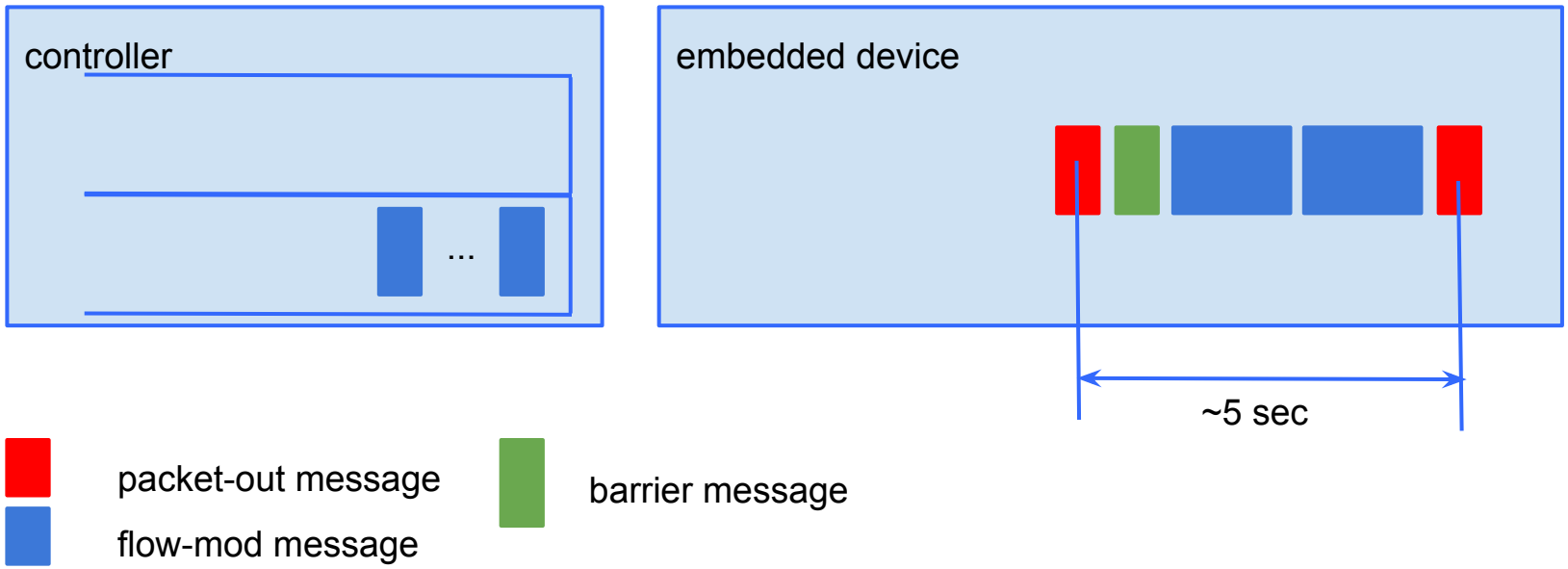
Time Sensitivity

Google



- Maintain separate queues
- Flow control using barriers: limit #outstanding flow-mod

Time Sensitivity



- Maintain separate queues
- Flow control using barriers: limit #outstanding flow-mod
  - TCP pipe will contribute to buffering
- Better interleaving due to prioritization

OFC Failover

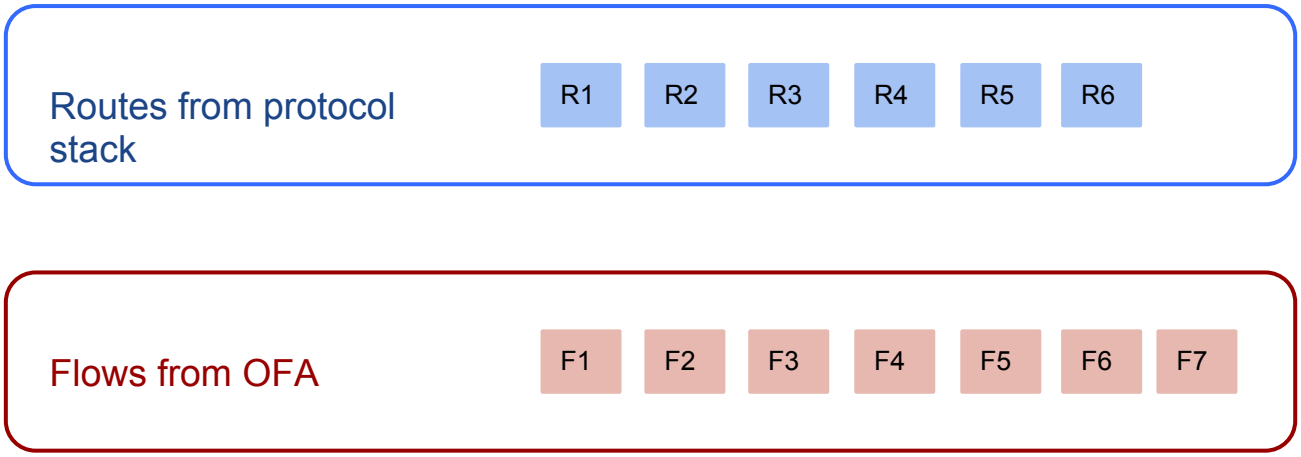
Google

- Master/Slave OFC for fault tolarence
- After failover new master needs to rebuild state
  - Gets **all** routes from protocol stack
  - Gets **all** flows from OFA
  - How to reconcile without impacting traffic?

OFC Failover

Google

- Master/Slave OFC for fault tolarence
- After failover new master needs to rebuild state
  - Gets **all** routes from protocol stack
  - Gets **all** flows from OFA
  - How to reconcile without impacting traffic?

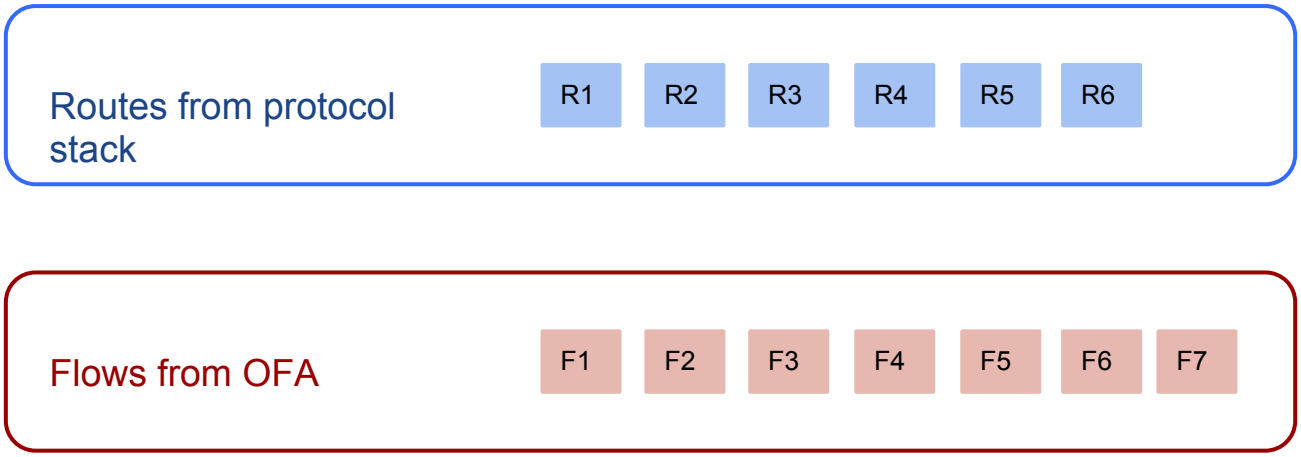


OFC Failover

Google

- Master/Slave OFC for fault tolarence
- After failover new master needs to rebuild state
  - Gets **all** routes from protocol stack
  - Gets **all** flows from OFA
  - How to reconcile without impacting traffic?

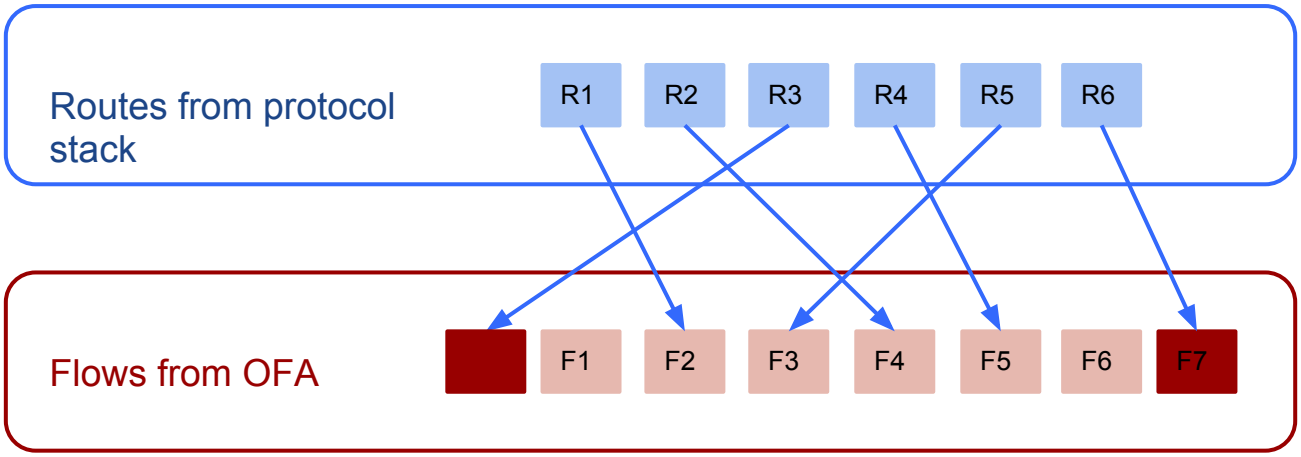
Authoritative



OFC Failover

Google

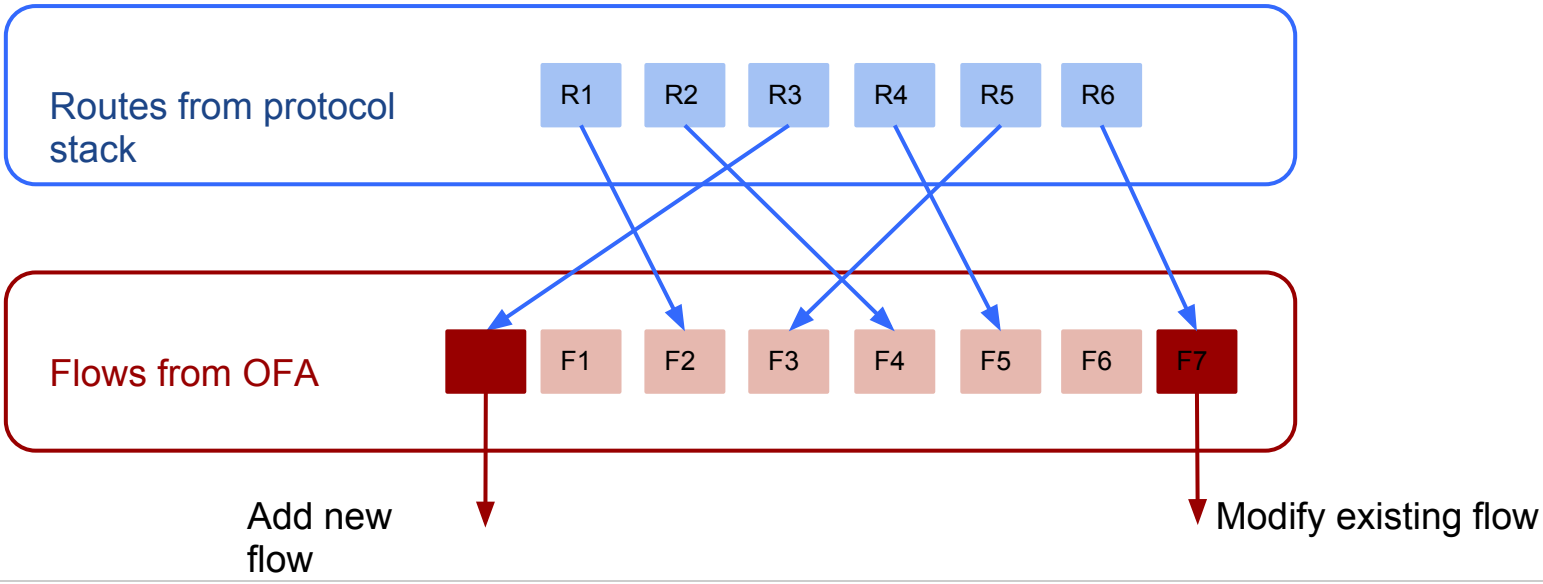
- Master/Slave OFC for fault tolarence
- After failover new master needs to rebuild state
  - Gets **all** routes from protocol stack
  - Gets **all** flows from OFA
  - How to reconcile without impacting traffic?



OFC Failover

Google

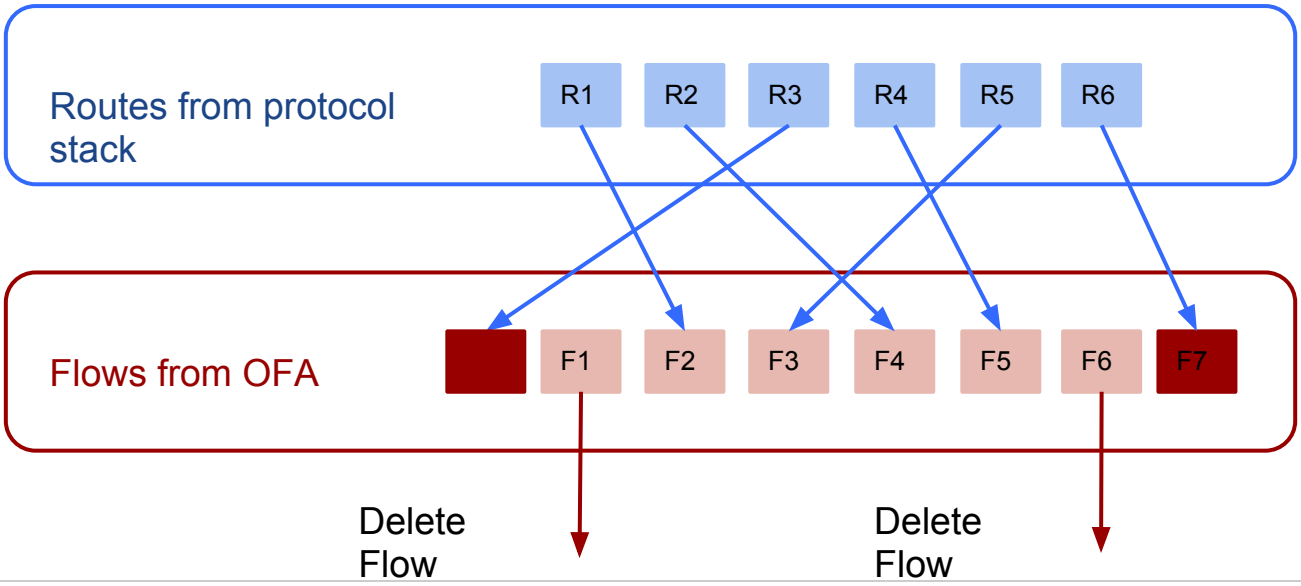
- Master/Slave OFC for fault tolarence
- After failover new master needs to rebuild state
  - Gets **all** routes from protocol stack
  - Gets **all** flows from OFA
  - How to reconcile without impacting traffic?



OFC Failover

Google

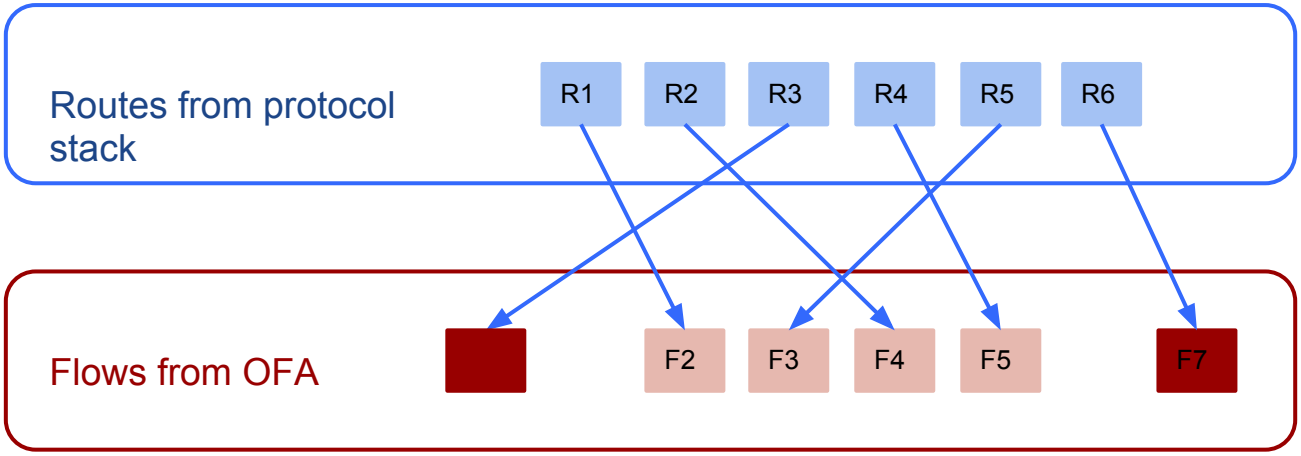
- Master/Slave OFC for fault tolarence
- After failover new master needs to rebuild state
  - Gets **all** routes from protocol stack
  - Gets **all** flows from OFA
  - How to reconcile without impacting traffic?



OFC Failover

Google

- Master/Slave OFC for fault tolarence
- After failover new master needs to rebuild state
  - Gets **all** routes from protocol stack
  - Gets **all** flows from OFA
  - How to reconcile without impacting traffic?



OFC Failover

Google

- Master/Slave OFC for fault tolarence
- After failover new master needs to rebuild state
  - Gets **all** routes from protocol stack
  - Gets **all** flows from OFA
  - How to reconcile without impacting traffic?
    - What if ARPs are not resolved?
    - What if flows use indirect nexthops/groups?
    - How to scale this for many routers?

Thank you!

Google™

Questions?