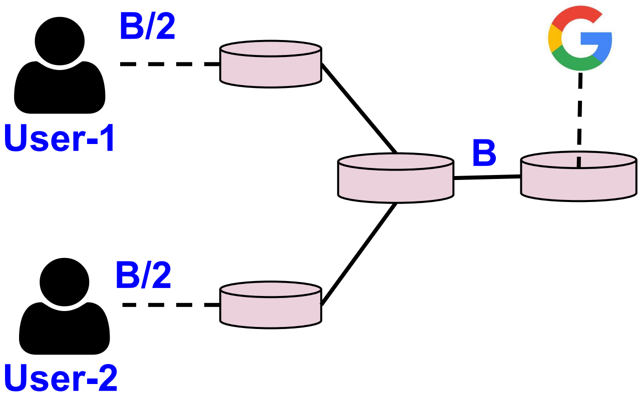
Network Resource Management as a Database Problem

(Vision Paper)

Hafiz Mohsin Bashir
Abdullah Bin Faisal, Fahad R. Dogar

Network Resource Management

Network resource management: How to efficiently share **network bandwidth** amongst the users/apps?



TCP: A classic example

- **Fair-Share policy*:***
  - Contenders equally share link BW
- **End-host based distributed mechanism**
  - Additive increase, multiplicative decrease
- **Tightly couple policy and mechanism**

Network Resource Management Inside a Cloud

Varying performance objectives

⇩

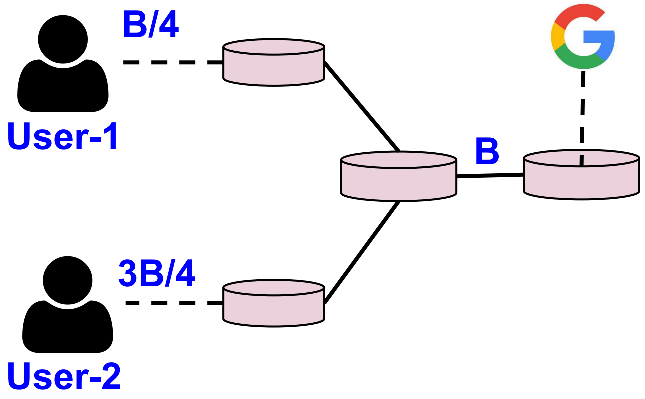
Need for rich set of policies beyond fairshare

Network Resource Management Inside a Cloud

Varying performance objectives

⇩

Need for rich set of policies beyond fairshare



Example
Providing Bandwidth Guarantee

Network Resource Management Inside a Cloud

Varying performance objectives

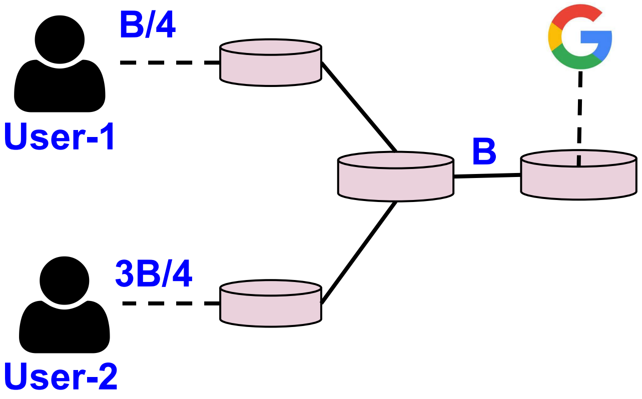
⇩

Need for rich set of policies beyond fairshare

⇩

D3 SIGCOMM ‘11	PDQ SIGCOMM ‘12	PASE SIGCOMM ‘14	BARAAT SIGCOMM ‘14
PIAS NSDI ‘15	2D CoNEXT ‘18	DAS CoNEXT ‘19	Varys SIGCOMM ‘14

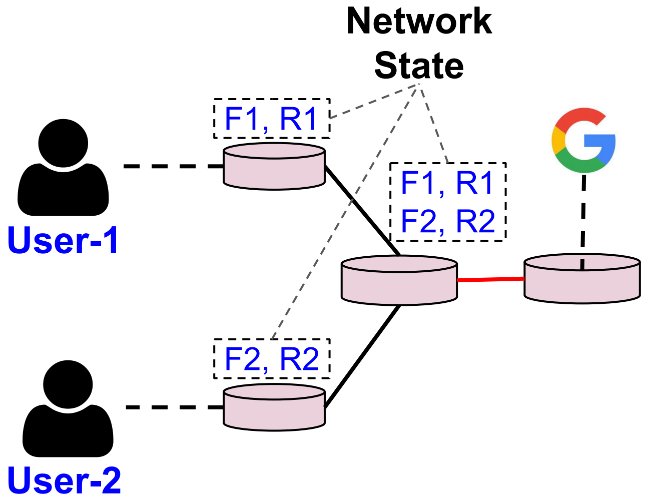
A large body of work



Example
Providing Bandwidth Guarantee

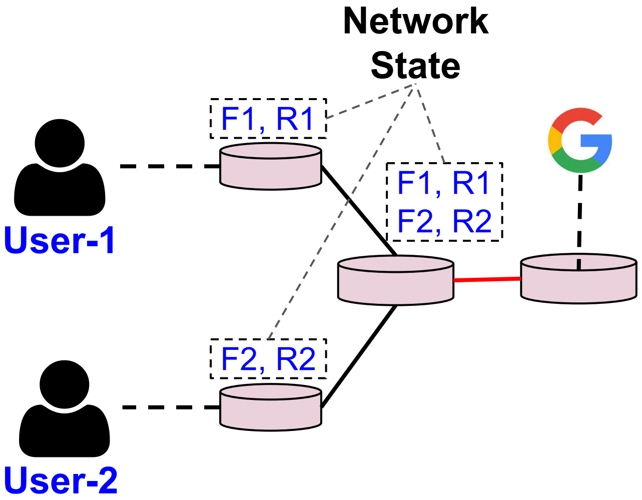
Limitations of Existing Approaches

- **Distributed approach**
  - **Limited Control:** State is distributed across nodes
  - **Complex:** Requires coordination between nodes



Limitations of Existing Approaches

- **Distributed approach**
  - **Limited Control:** State is distributed across nodes
  - **Complex:** Requires coordination between nodes
- **Point Solutions**
  - **Limits support for future use-cases**
  - **Scalability Challenge:** Infrastructure evolves
  - **Coexistence Challenge:** Point solutions are hard to co-exist



"Tying congestion control deeply to switch internals poses a larger maintenance burden (e.g., finding appropriate thresholds)"

Google LLC SIGCOMM '20

A case for a Centralized Approach: Inspired by SDN

- **Control over network state**
    - **Opportunity:** Greater control over the network resources
    - **Challenge:** Make it scalable

A case for a Centralized Approach: Inspired by SDN

- **Control over network state**
  - **Opportunity:** Greater control over the network resources
  - **Challenge:** Make it scalable

- **Decoupling of policy and mechanism**
  - **Opportunity:** A set of key parameters can enable many policies
  - **Challenge:** Provide efficient enforcement mechanism

A case for a Centralized Approach: Inspired by SDN

- **Control over network state**
  - **Opportunity:** Greater control over the network resources
  - **Challenge:** Make it scalable

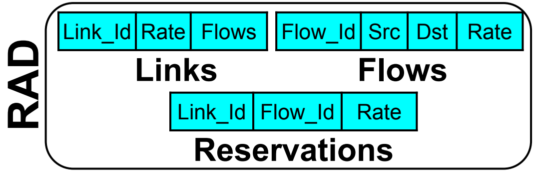
- **Decoupling of policy and mechanism**
  - **Opportunity:** A set of key parameters can enable many policies
  - **Challenge:** Provide efficient enforcement mechanism

- **Abstractions:** Support for a variety of use-cases
  - **Opportunity:** Build new abstractions on top of centralized state
  - **Challenge**: Identify suitable abstractions

A Database Abstraction For Network Resource Management

Resource Allocation Database (RAD): An Overview

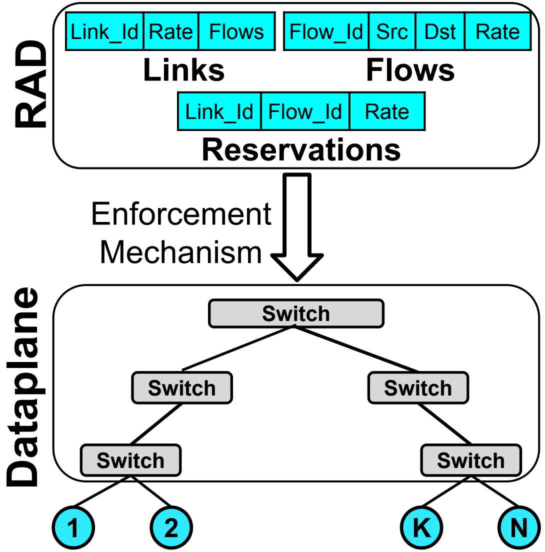
- **Resource management**
  - **Centralization:** Database tables store the state
  - **Control:** Manage bandwidth sharing decisions



Resource Allocation Database (RAD): An Overview

- **Resource management**
  - **Centralization:** Database tables store the state
  - **Control:** Manage bandwidth sharing decisions

- **Decoupling policy from mechanism**
  - An efficient mechanism reflects database state onto the switches

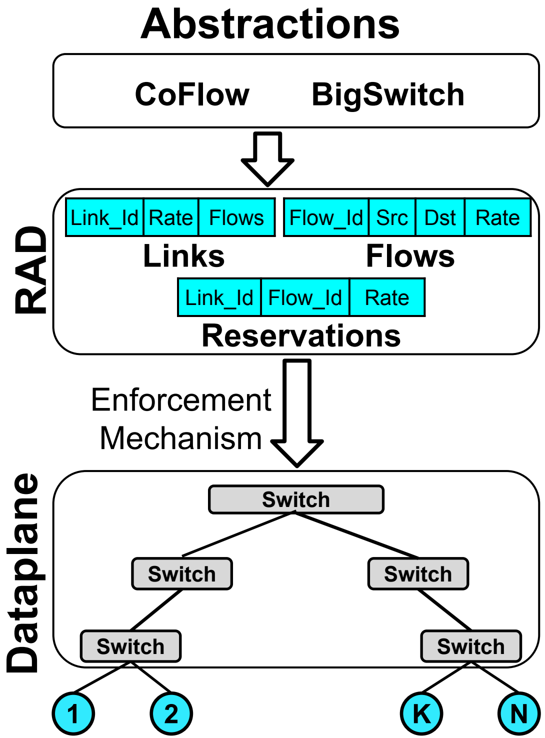


Resource Allocation Database (RAD): An Overview

- **Resource management**
  - **Centralization:** Database tables store the state
  - **Control:** Manage bandwidth sharing decisions

- **Decoupling policy from mechanism**
  - An efficient mechanism reflects database state onto the switches

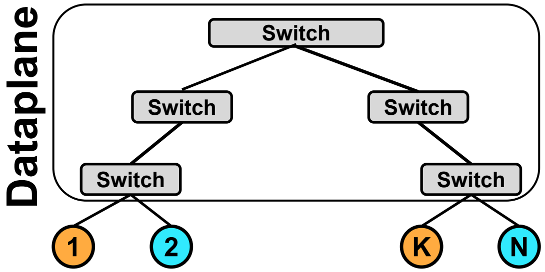
- **Abstractions:**
  - Builds on top of RAD tables
  - Represents different use cases



Use Case - Bandwidth Reservation

- Fundamental to provide guaranteed service (e.g., Baraat, PDQ, D3)

- **Classic Approach:**
    - **Establish consensus:** Multi-step process
    - **Atomicity:** Only reserve if all nodes can commit



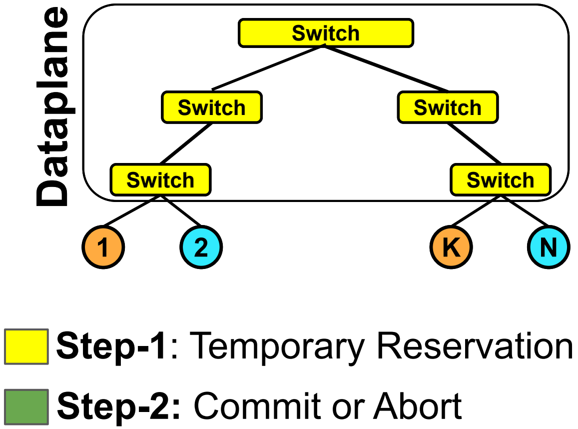
Step-1: Temporary Reservation

Step-2: Commit or Abort

Use Case - Bandwidth Reservation

- Fundamental to provide guaranteed service (e.g., Baraat, PDQ, D3)

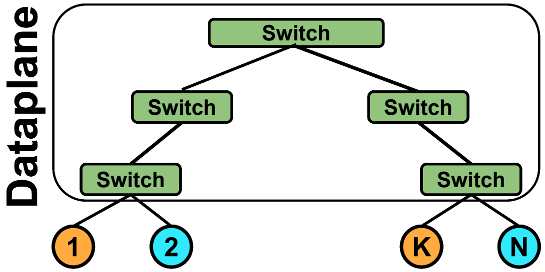
- **Classic Approach:**
  - **Establish consensus:** Multi-step process
  - **Atomicity:** Only reserve if all nodes can commit



Use Case - Bandwidth Reservation

- Fundamental to provide guaranteed service (e.g., Baraat, PDQ, D3)

- **Classic Approach:**
  - **Establish consensus:** Multi-step process
  - **Atomicity:** Only reserve if all nodes can commit



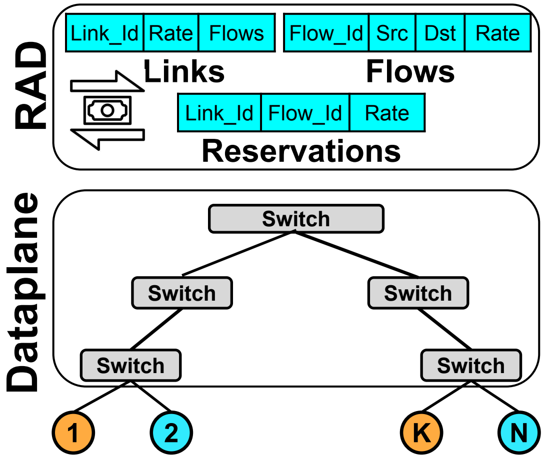
Step-1: Temporary Reservation

Step-2: Commit or Abort

Use Case - Bandwidth Reservation

- Fundamental to provide guaranteed service (e.g., Baraat, PDQ, D3)

- **Classic Approach:**
  - **Establish consensus:** Multi-step process
  - **Atomicity:** Only reserve if all nodes can commit



- **Using RAD: Database *transactions***
  - **Opportunity!** Built in support for atomic operations
  - **Challenge!** How to minimize the overhead of distributed transactions?

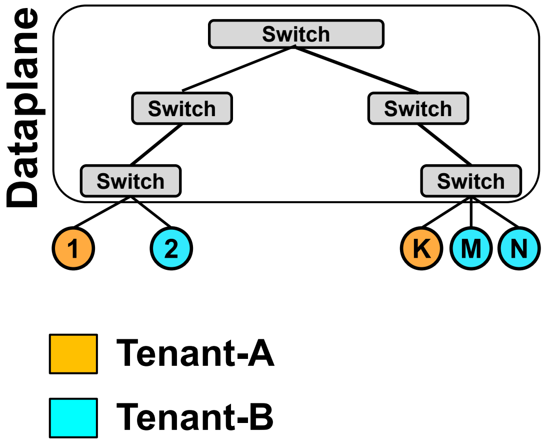
Use Case - Virtual View of the Network

- Common requirement inside a cloud
  (e.g., BigSwitch, Virtual Cluster)

Use Case - Virtual View of the Network

- Common requirement inside a cloud (e.g., BigSwitch, Virtual Cluster)

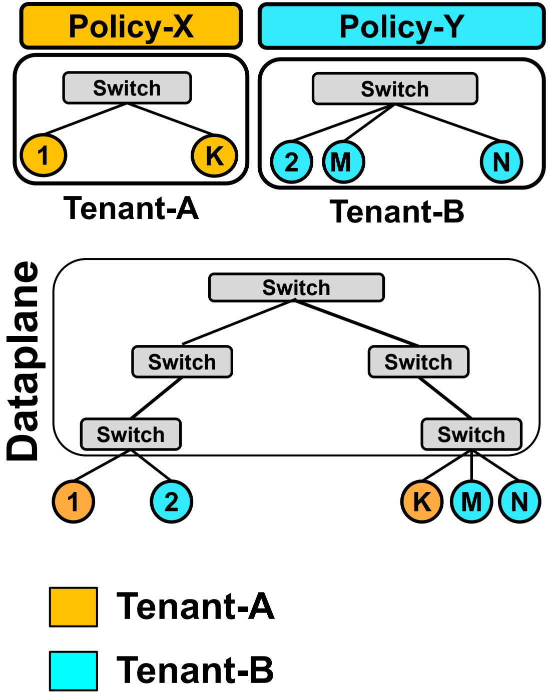
- ***Virtual private cluster*** **(e.g., multi-tenant setting)**
  - Independent control over allocated resources



Use Case - Virtual View of the Network

- Common requirement inside a cloud (e.g., BigSwitch, Virtual Cluster)

- ***Virtual private cluster*** **(e.g., multi-tenant setting)**
  - Independent control over allocated resources

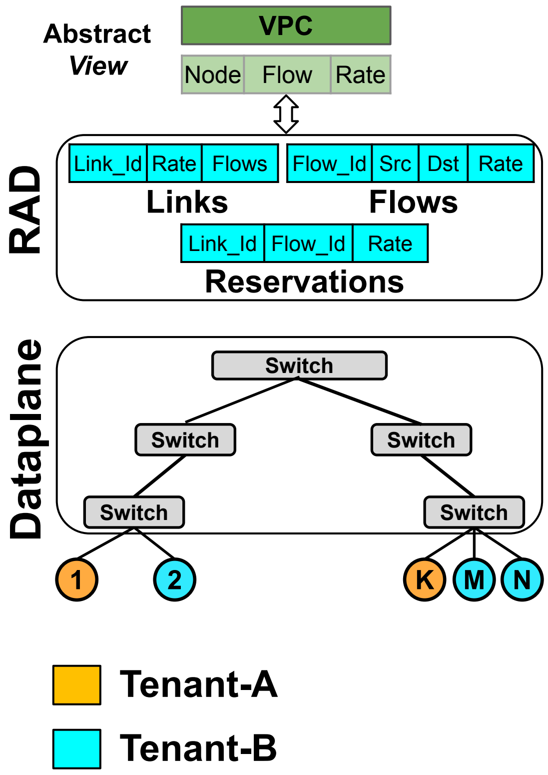


Use Case - Virtual View of the Network

- Common requirement inside a cloud (e.g., BigSwitch, Virtual Cluster)

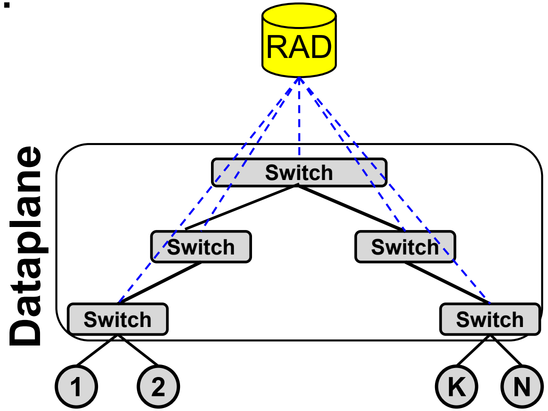
- ***Virtual private cluster*** **(e.g., multi-tenant setting)**
  - Independent control over allocated resources

- **Using database *views***
  - **Opportunity!** Natural fit for data independence
  - **Challenge!** Can *views* be made updateable?



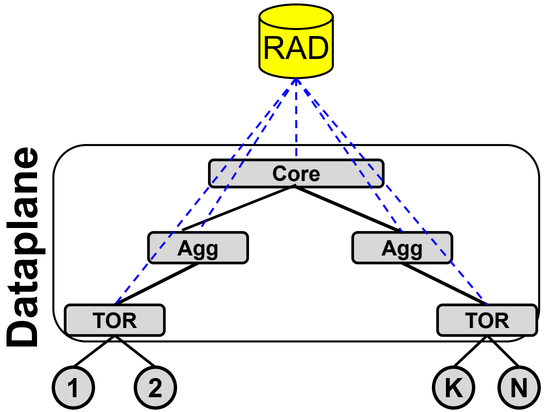
Challenges in Realizing RAD in Large Data Centers

- **Scalability:** Millions of requests per second?
- **Performance:** Minimize delays in accessing RAD?



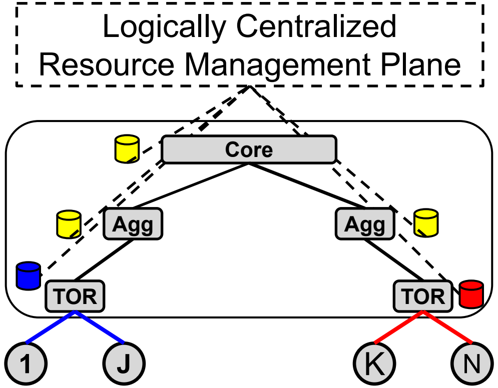
Scalability and Performance Concerns

- **Data Center network properties used by RAD**
  - **Topologies:** Structured like a tree
  - **Traffic locality:** Majority of the traffic is rack-local



Scalability and Performance Concerns

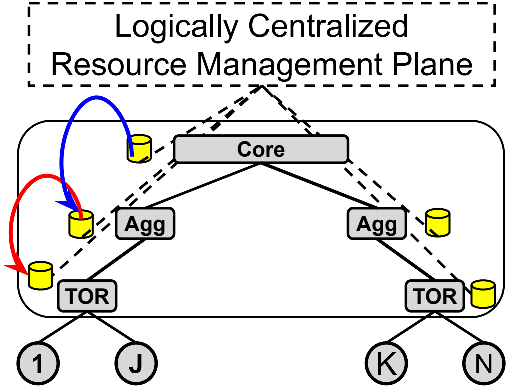
- **Data Center network properties used by RAD**
  - **Topologies:** Structured like a tree
  - **Traffic locality:** Majority of the traffic is rack-local



- **Opportunity! Network Aware Sharding**
  - Shard network links across RAD instances
  - Each switch has a co-located RAD instance
  - Rack-local queries only contact local RAD replica

Scalability and Performance Concerns

- **Data Center network properties used by RAD**
  - **Topologies:** Structured like a tree
  - **Traffic locality:** Majority of the traffic is rack-local



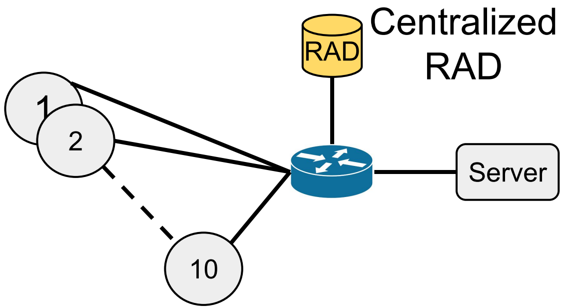
- Opportunity! **Network aware replication**
  - A switch only replicates to its children (e.g., Core->Agg, Agg->TOR)
  - Choice over consistency guarantees across replicas

Preliminary Evaluation: Overheads of RAD

- **Objective:** Evaluate the feasibility of using RAD

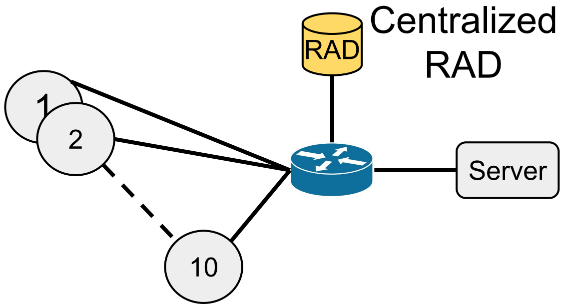
Preliminary Evaluation: Overheads of RAD

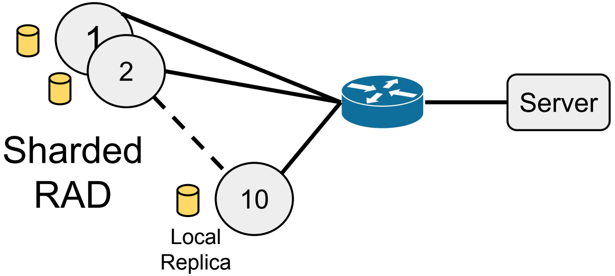
- **Objective:** Evaluate the feasibility of using RAD
- **Setup:**
  - **Topology**: 10 clients, 1 server, 1 RAD node
  - **Database**: Off-the-shelf Mysql
  - **Workload:** Websearch
  - **Policy:** Fairshare (TCP)
  - **Metric**: Flow completion time (FCT)



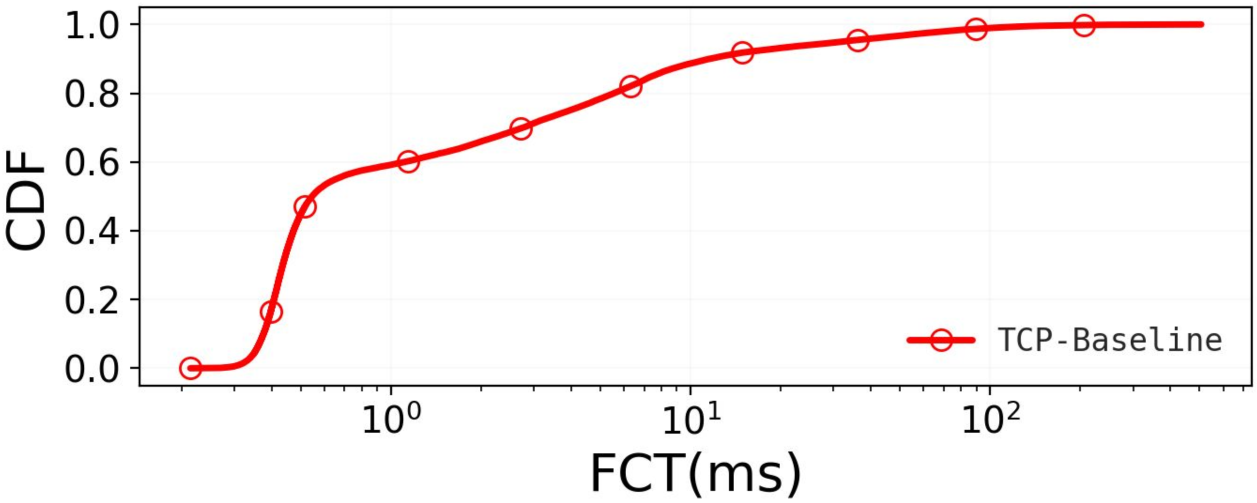
Preliminary Evaluation: Overheads of RAD

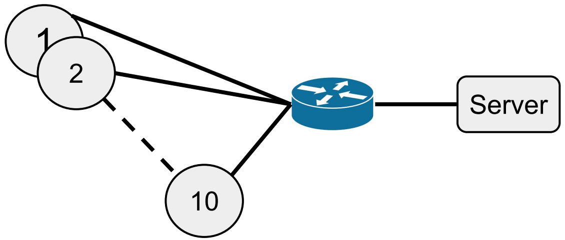
- **Objective:** Evaluate the feasibility of using RAD
- **Setup:**
  - **Topology**: 10 clients, 1 server, 1 RAD node
  - **Database**: Off-the-shelf Mysql
  - **Workload:** Websearch
  - **Policy:** Fairshare (TCP)
  - **Metric**: Flow completion time (FCT)
- **Schemes:**
  - **TCP:** Baseline (Does not use database)
  - **RAD:** Single centralized DB
  - **RAD-Sharded:** Each client has a local copy of RAD



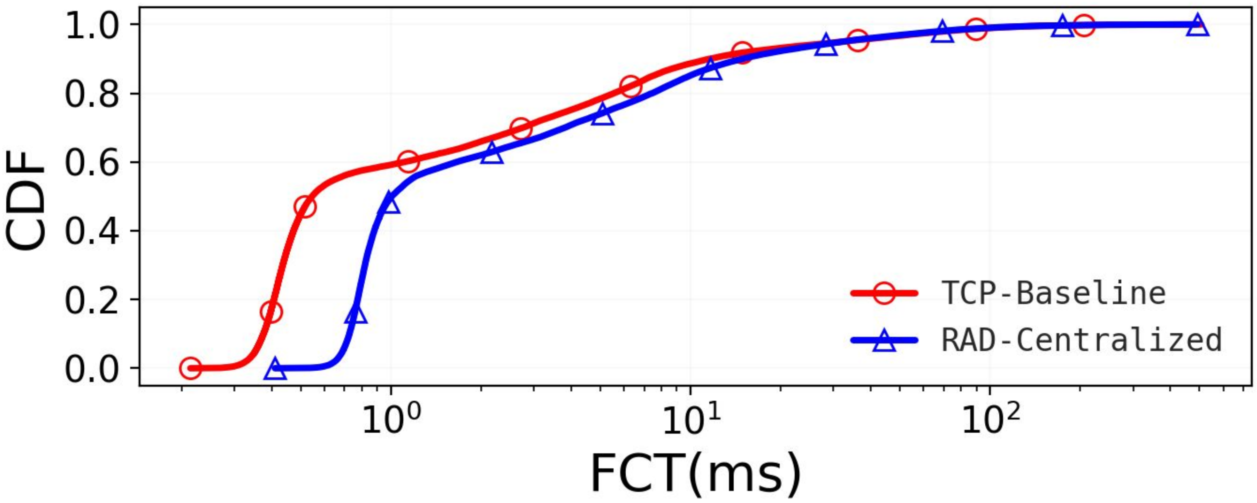


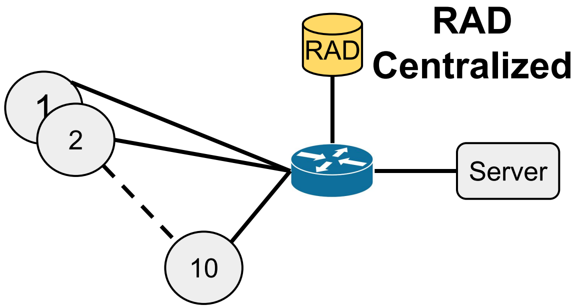
Preliminary Evaluation:



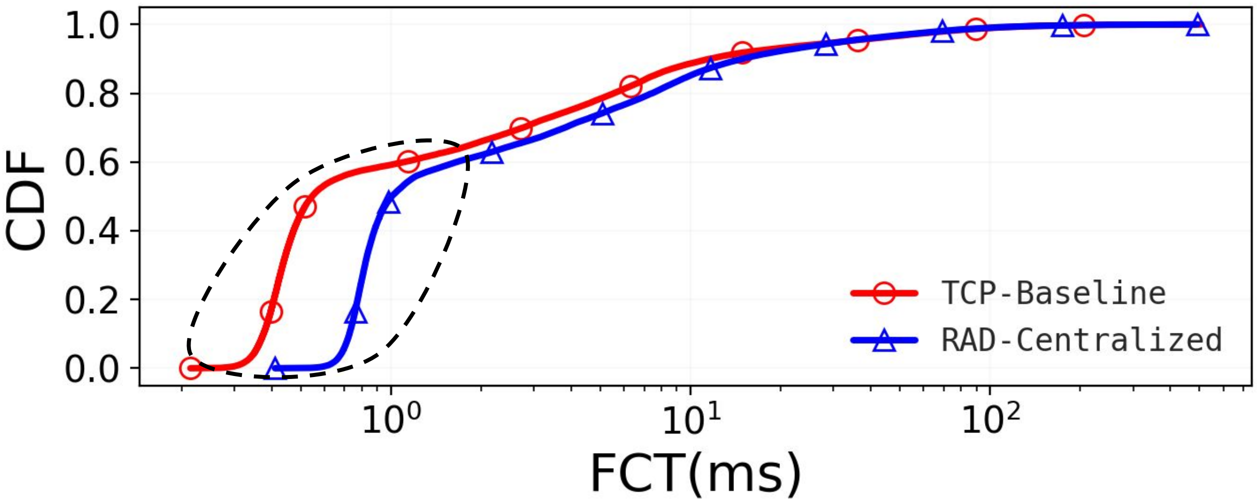


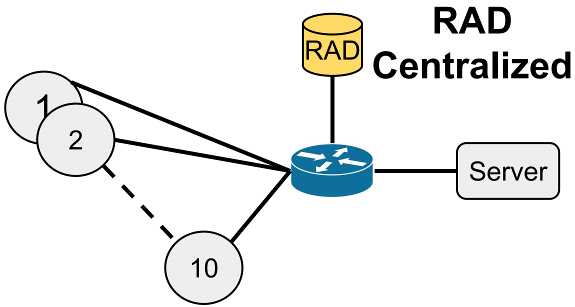
Preliminary Evaluation: Feasibility of RAD





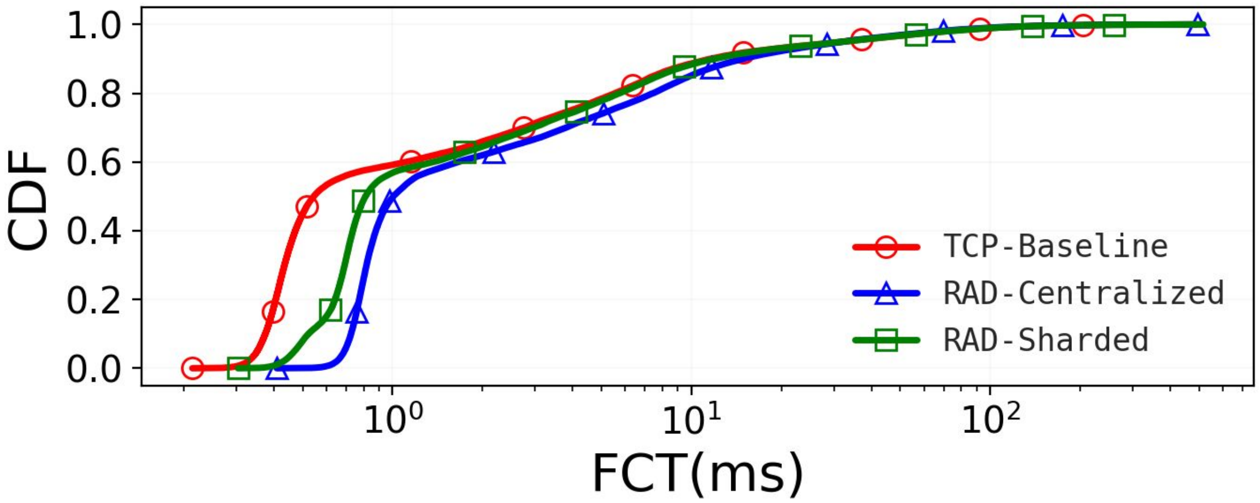
Preliminary Evaluation: Feasibility of RAD

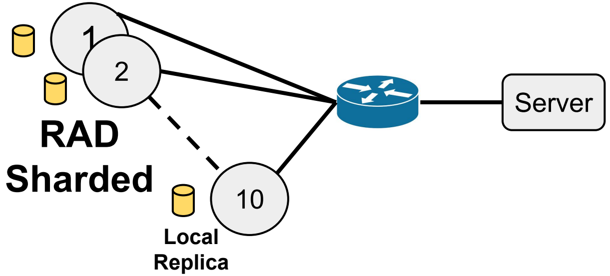




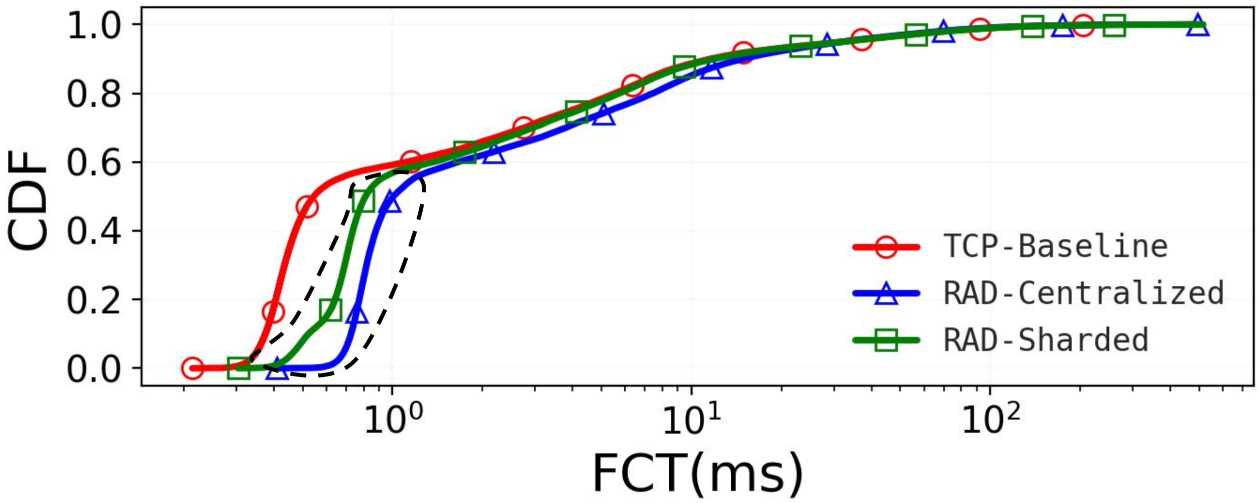
Upto 60% flows experience significant additional latency

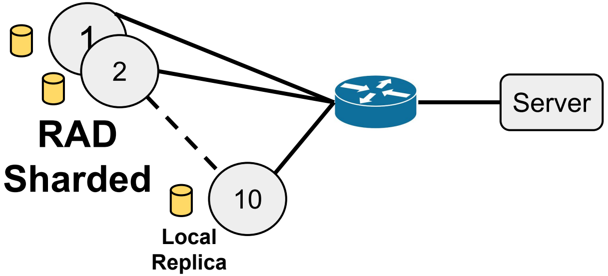
Preliminary Evaluation: Overheads of RAD





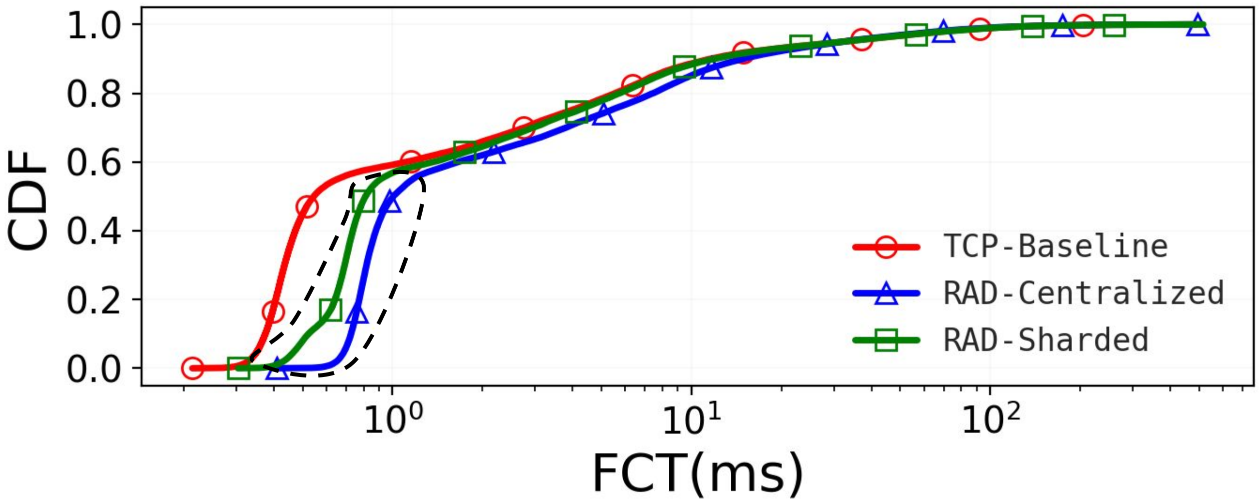
Preliminary Evaluation: Overheads of RAD

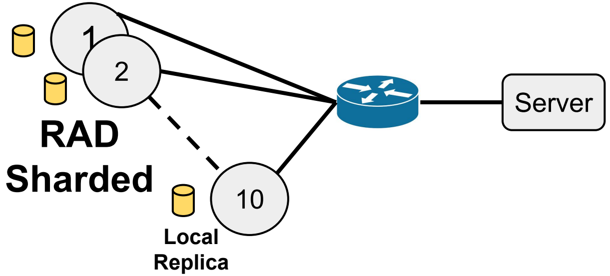




Sharding helps in reducing RAD overhead

Preliminary Evaluation: Overheads of RAD





Sharding helps in reducing RAD overhead

Question! Can we do better?

Summary

- Network resource management inside a cloud is a complex task
- A database approach inspired by SDN is promising
  - Simplifies the network resource management task
  - Interesting challenges to take care
- Opens avenues for exciting research

Questions?

Extra Slides

Implementation

- **Database support**
  - **Mysql off-the-shelf**
  - **Caching:** Many flow requests are identical from DB perspective

- **End host rate control:**
  - **Modified TCP stack:** Added bound support on TCP window

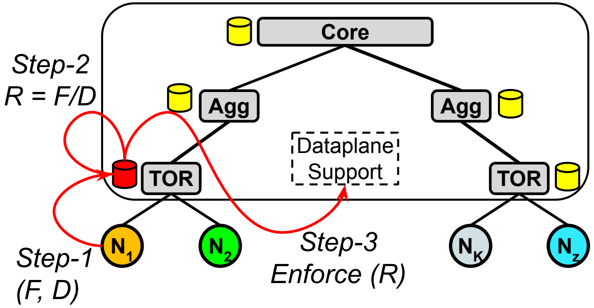
- **Traffic Generation:** trafficGenerator from HKUST-SING Lab

Technical Challenges and Opportunities

- **Distributed Transaction**
  - Typically a high latency operation
  - Recent advances (e.g., RDMA) can help
- **Replication Overhead**
  - A suitable consistency models can help lower the overheads

End-to-End example

- **Rate reservation**
  - **Given**: A flow size, deadline
  - **Action:** Allocate a suitable rate to meet the deadline



Setup: N_1 wants to send data of size F in time D to N_2

Client: Node-1 initiates a new flow request and sends flow size (F) and deadline info (D) to local RM replica

RM: Calculates required rate (R = F/D) and update the state or rejects the request

RM: Enforces the rate (R) into the data plane on flows path

Other Use Cases

- **Accuracy VS Overhead Tradeoff**
  - Different use cases may require different level of accuracy
  - **RAD:** Various consistency models

- **Resource Management Sandboxing**
  - A scheme may require sandboxing to tune/optimize various parameters
  - **RAD:** Checkpointing can help replay events

Fairsharing over BigSwitch Abstraction

BigSwitch		
Node	Flow_Count	Rate

```
SELECT flow_count INTO _count FROM bigSwitch
WHERE node=new.dst;
IF (_count = 0) THEN SET _rate=maxRate
ELSE SET _rate=maxRate/_count+1
END IF;
CALL UPDATE_FLOW_RATES(new.dst, _rate);
```

PathLinks		
Path_Id	Link_Id	Seq

LinkState		
Link_Id	Rate	Flow_Count

PathMap		
Path_Id	Src	Dst

Flows				
Flow_Id	Src	Dst	Rate	Status

```
CREATE VIEW    BigSwitch
AS SELECT      DISTINCT Src AS Node, Flow_Count, Rate
FROM           PathMap
JOIN           PathLinks ON PathMap.Path_ID=PathLinks.Path_ID
JOIN           LinkState ON PathLinks.Link_ID=LinkState.Link_ID
WHERE          PathLinks.Seq=0;
```

Abstractions: Ravel vs RAD

BigSwitch		
Node_ID	Flow_Count	Rate

Flows			
Flow_ID	Src	Dst	Rate

PathLinks		
Path_ID	Link_ID	Link_Seq

LinkState		
Link_ID	Flow_Count	Rate

PathMap		
Path_ID	Src	Dst

FlowState					
Flow_ID	CoFlow_ID	Src	Dst	Rate	Status

```
CREATE VIEW   BigSwitch
AS SELECT     DISTINCT Src AS Node_ID, Flow_Count
FROM          PathMap
JOIN          Paths ON Pathmap.Path_ID=Paths.Path_ID
JOIN          Links ON Paths.Link_ID=Links.Link_ID
WHERE         Paths.Link_Seq=0;
```

Policy: FIFO

```
CREATE TRIGGER Policy_FIFO

BEFORE INSERT ON event
FOR EACH ROW
BEGIN
  DECLARE _flow_count INT(11);
  DECLARE _rate INT(11);
  SELECT max(flow_count) INTO _flow_count FROM BigSwitch WHERE node=new.src OR node=new.dst;
  IF (_flow_count > 0) THEN
    SET _rate = 0
  ELSE
    SET _rate = 1000
  END IF;
  INSERT INTO flows(flow_id, src, dst, rate) VALUES(new.flow_id, new.src, new.dst, _rate);
  UPDATE BigSwitch SET flow_count=flow_count+1 WHERE Node=new.src OR Node=new.dst;

BEFORE DELETE ON event
FOR EACH ROW
BEGIN
  DECLARE _min_id INT(11);
  DECLARE _flow_id INT(11);
  DELETE FROM flows WHERE flow_id=new.flow_id;
  UPDATE BigSwitch SET flow_count=flow_count-1 WHERE Node=new.src OR Node=new.dst;
  SELECT MIN(seq), flow_id INTO _flow_id FROM flows;
  UPDATE flows SET rate=1000 WHERE flow_id=_flow_id;
END&&
```