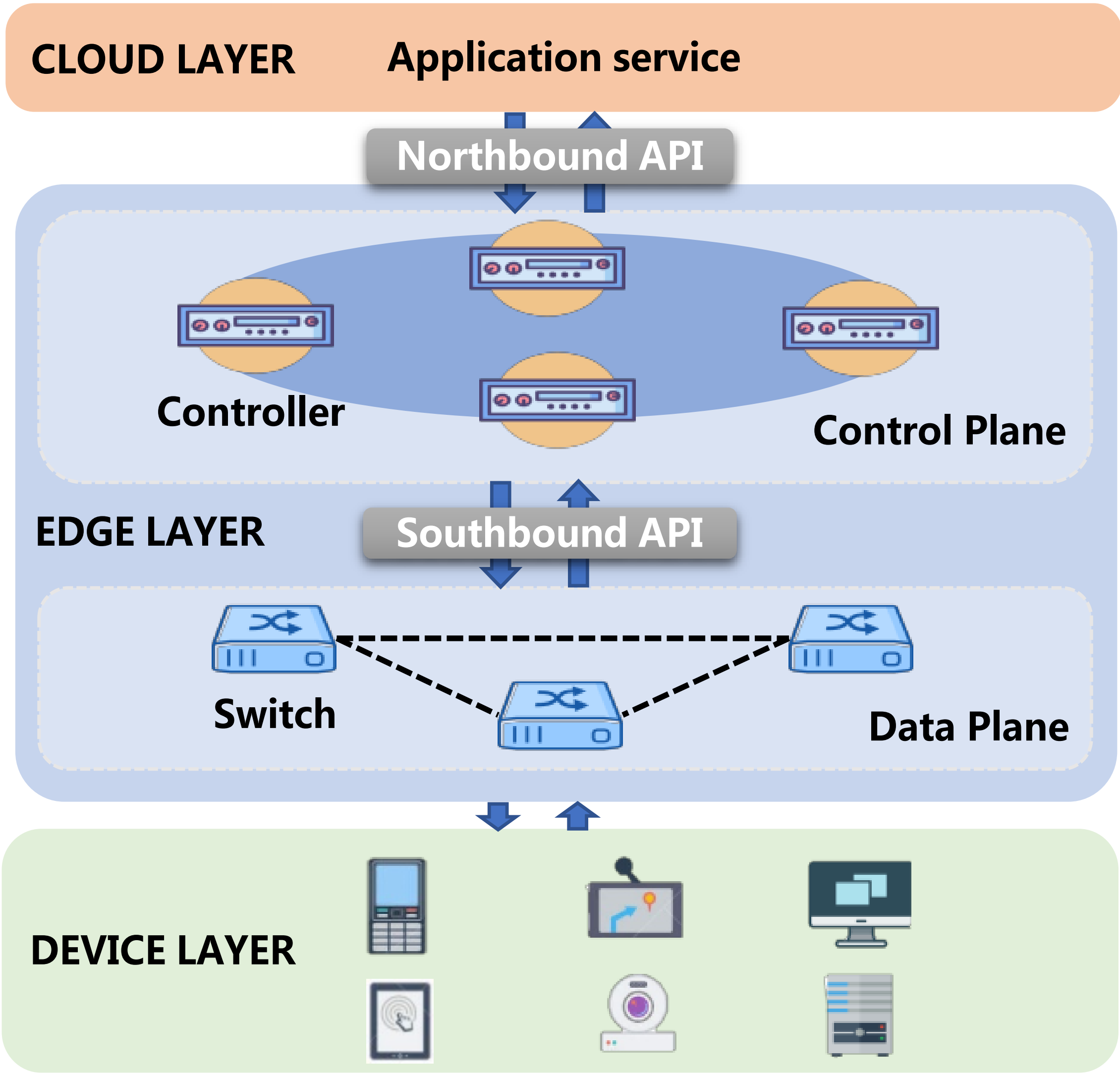
Curb: Trusted and Scalable Software-Defined Network Control Plane for Edge Computing

Minghui Xu[#], Chenxu Wang[#], Yifei Zou[#], Dongxiao Yu[#], Xiuzhen Cheng[#] and Weifeng Lyu[*]

[#] Shandong University
[*] Beihang University

July 9, 2022

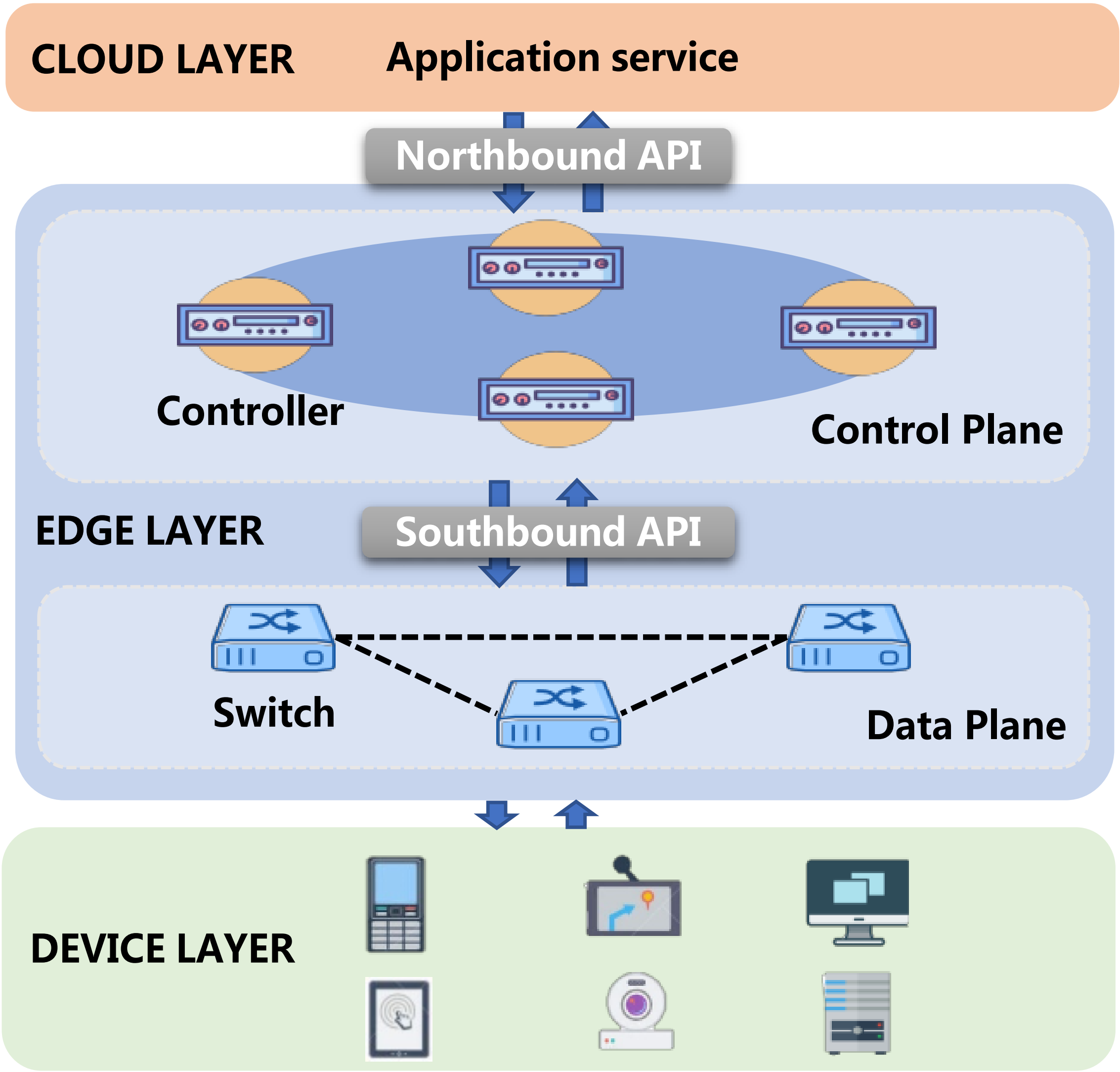
Background



Software defined network (SDN)

- ✓ Decouple control and data plane
- ✓ Open-programming interfaces

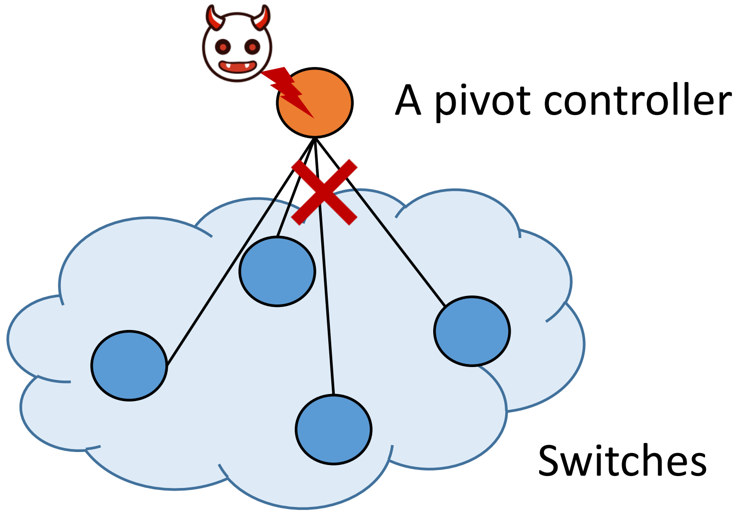
Background



Software defined network (SDN)

- ✓ Decouple control and data plane
- ✓ Open-programming interfaces

Single point of failure



Related work

Techniques	Papers
Primary-backup control plane	Morph: An adaptive framework for efficient and byzantine fault-tolerant sdn control plane, JSAC, 2018
	Byzantine-besilient controller mapping and remapping in software defined networks, TNSE, 2020
Byzantine fault tolerance (BFT) consensus algorithm	Byzantine fault tolerant software-defined networking (sdn) controllers, COMPSAC, 2016
	Bft protocols for heterogeneous resource allocations in distributed sdn control plane, ICC, 2019
	P4bft: Hardware-accelerated byzantine-resilient network control plane, GLOBECOM, 2019
Blockchain	Information classification strategy for blockchain-based secure sdn in iot scenario, INFOCOM WKSHPS, 2020
	A blockchain-sdn-enabled internet of vehicles environment for fog computing and 5g networks, IoTJ, 2019

Related work

Primary-backup control plane

- Map each switch to f+1 primary controllers and f back-up ones to defend against f byzantine nodes.

Blockchain technique

- Provide some security properties for SDN:
  - ✓ Provable security
  - ✓ Immutability
  - ✓ Traceability
  - ✓ Transparency

BFT consensus algorithms

- Controllers exchange messages to reach an agreement on a valid decision.
  - ✓ Guarantee the state consistency between controllers.
  - ✓ Resist attacks from byzantine nodes.

Motivation

Can we design a both trusted and scalable SDN control plane for edge computing?

- For primary-backup control plane, maintaining consistent node states is still a problem to be solved.

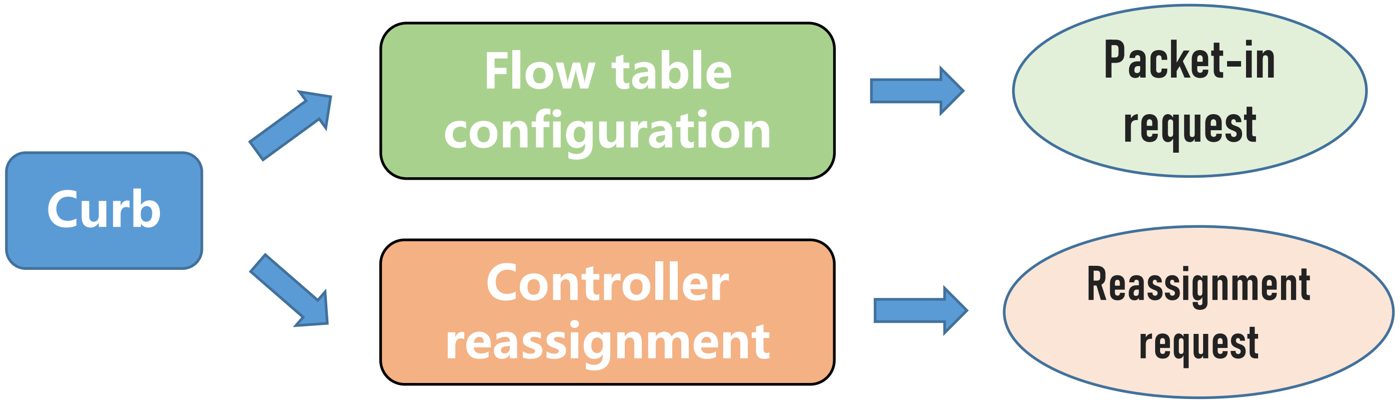
- Introducing BFT consensus incurs much communication overhead due to the need of massive message exchanges.

- Traditional blockchain systems have been criticized for their low throughput.

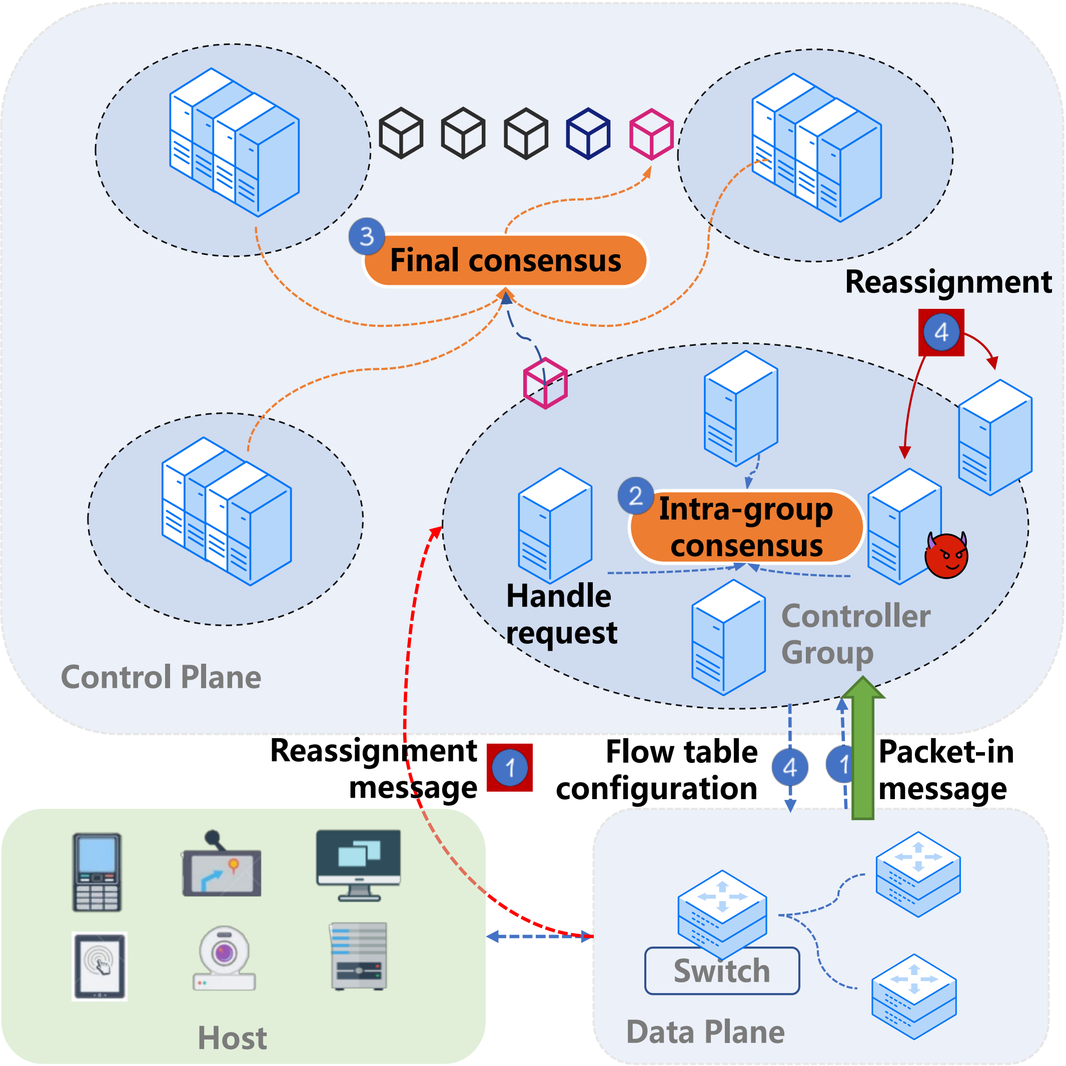
Contribution

- ✓ We propose Curb, a trusted and scalable SDN control plane on edge layer, which seamlessly incorporates blockchain and BFT consensus into group-based control plane, achieving byzantine fault tolerance, verifiability, consistency and scalability within one framework.
- ✓ Curb provides a blockchain-secured adaptive reassignment approach for SDN control plane. So byzantine controllers can be timely detected and then rapidly replaced with honest ones.
- ✓ Controllers are organized into multiple groups, each taking charge of multiple switches and reaching intra-group consensus in parallel. The message complexity of each round is reduced to O(N).

Functionalities of Curb



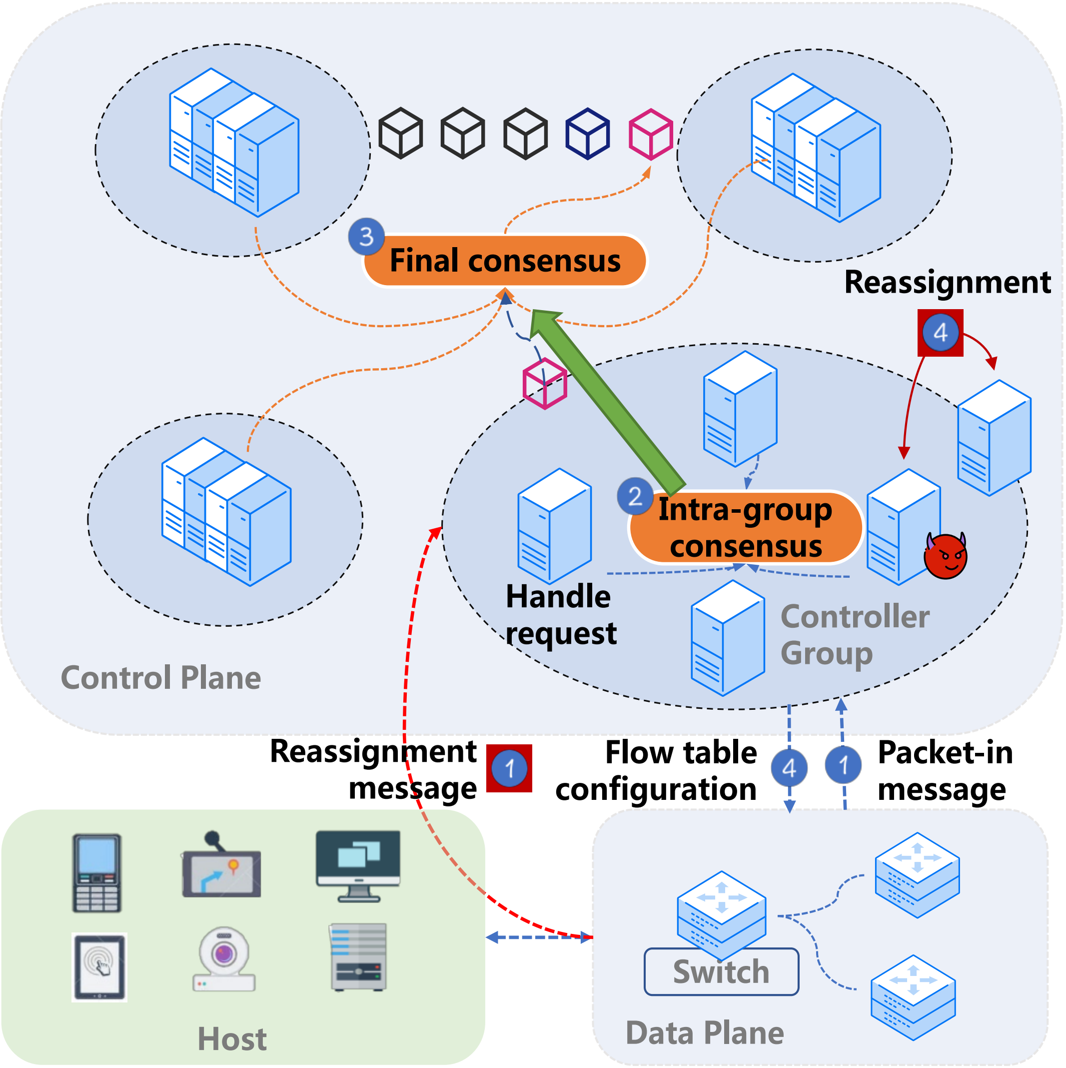
Workflow of Curb



Packet-in request

- **Step 0:** A user host sends a packet to the network so that it can be forwarded to its target host.
- **Step 1:** A switch sends a **PKT-IN** message to its assigned controller group to obtain forwarding rules.

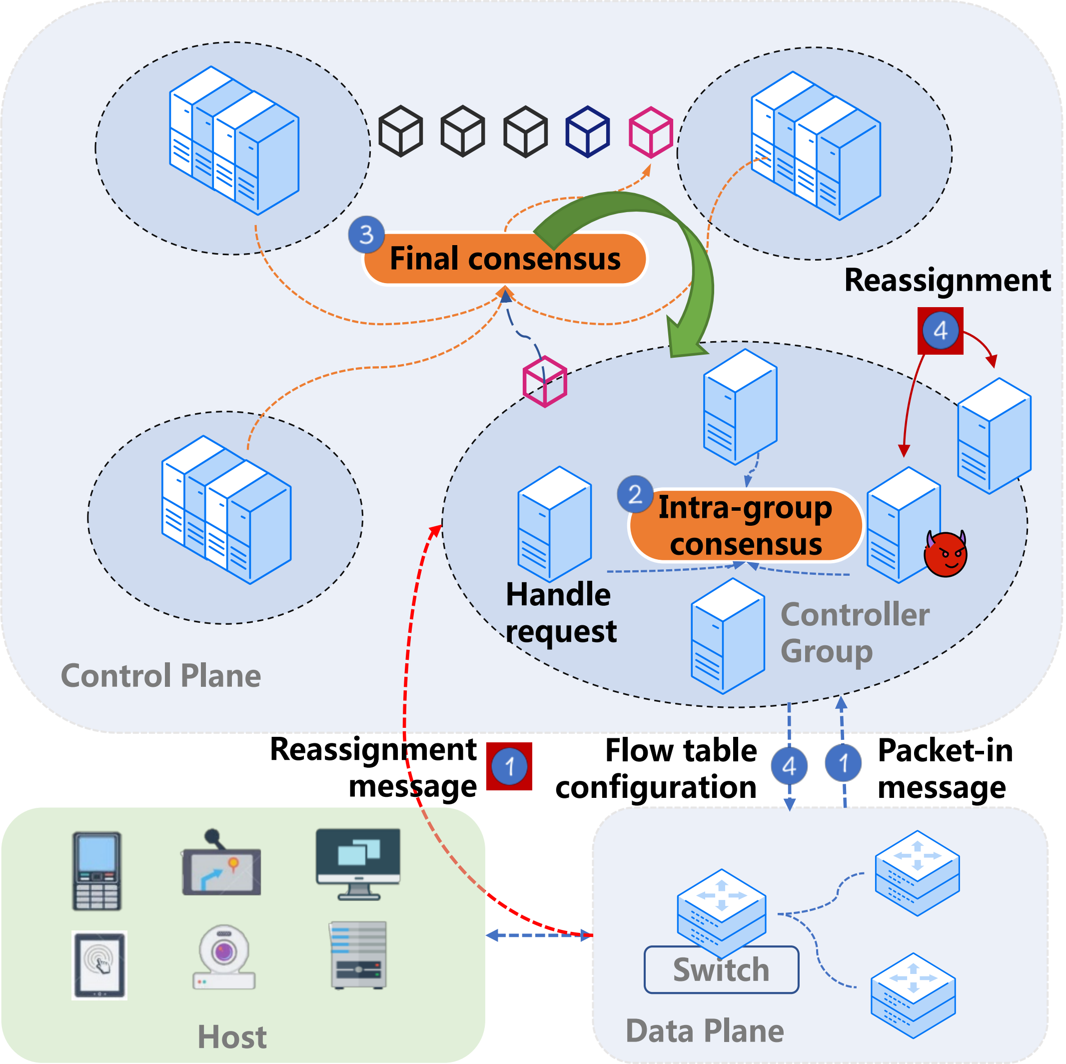
Workflow of Curb



Packet-in request

- **Step 0:** A user host sends a packet to the network so that it can be forwarded to its target host.
- **Step 1:** A switch sends a PKT-IN message to its assigned controller group to obtain forwarding rules.
- **Step 2:** The group members figure out forwarding rules and carry out the ***intra-group consensus*** process to reach consensus on the rules. After that they send blocks to the final committee.

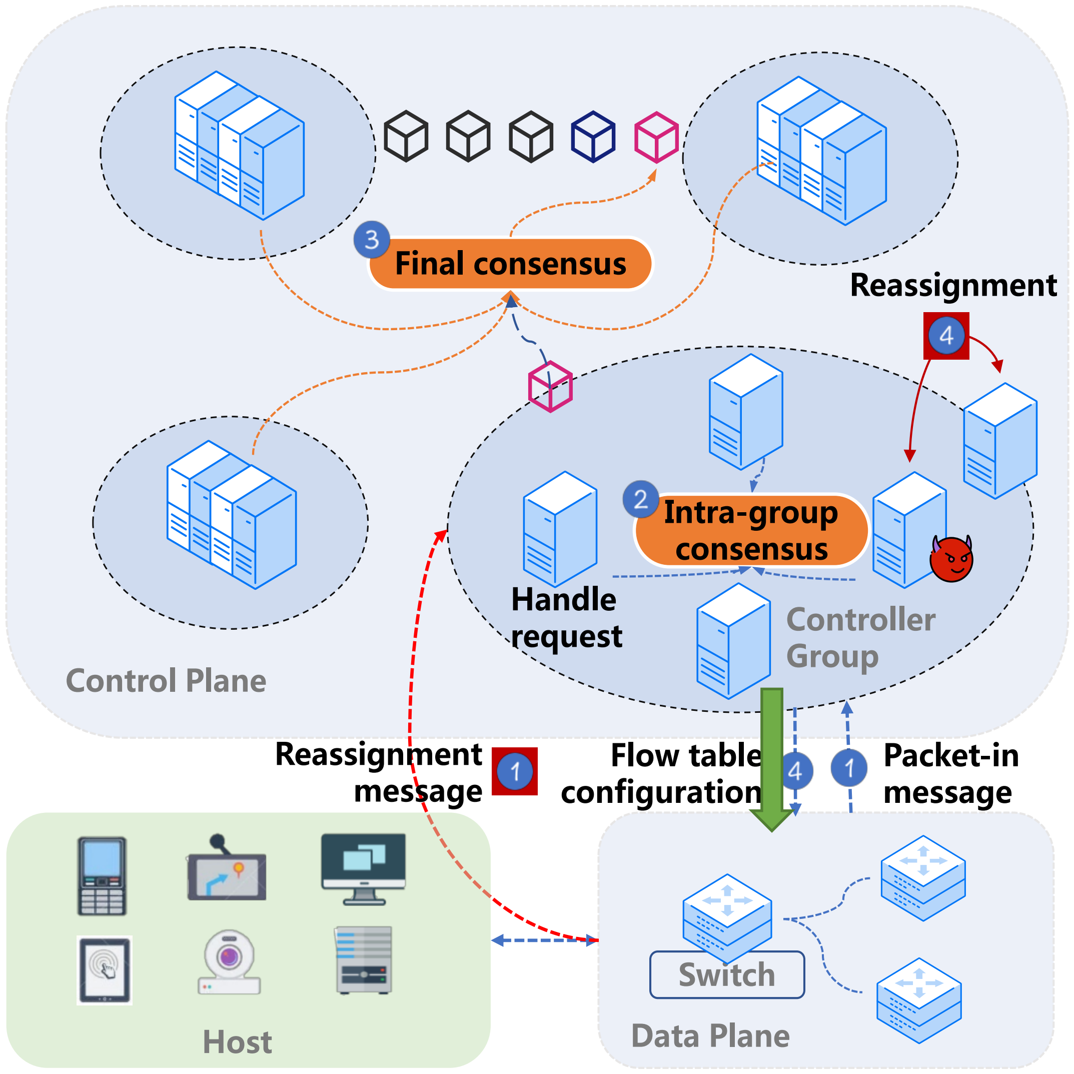
Workflow of Curb



Packet-in request

- **Step 0:** A user host sends a packet to the network so that it can be forwarded to its target host.
- **Step 1:** A switch sends a PKT-IN message to its assigned controller group to obtain forwarding rules.
- **Step 2:** The group members figure out forwarding rules and carry out the ***intra-group consensus*** process to reach consensus on the rules. After that they send blocks to the final committee.
- **Step 3:** The final committee takes charge of the ***final consensus*** process, where committee members reach consensus on blocks from multiple groups. After that the members broadcast blocks to every controller.

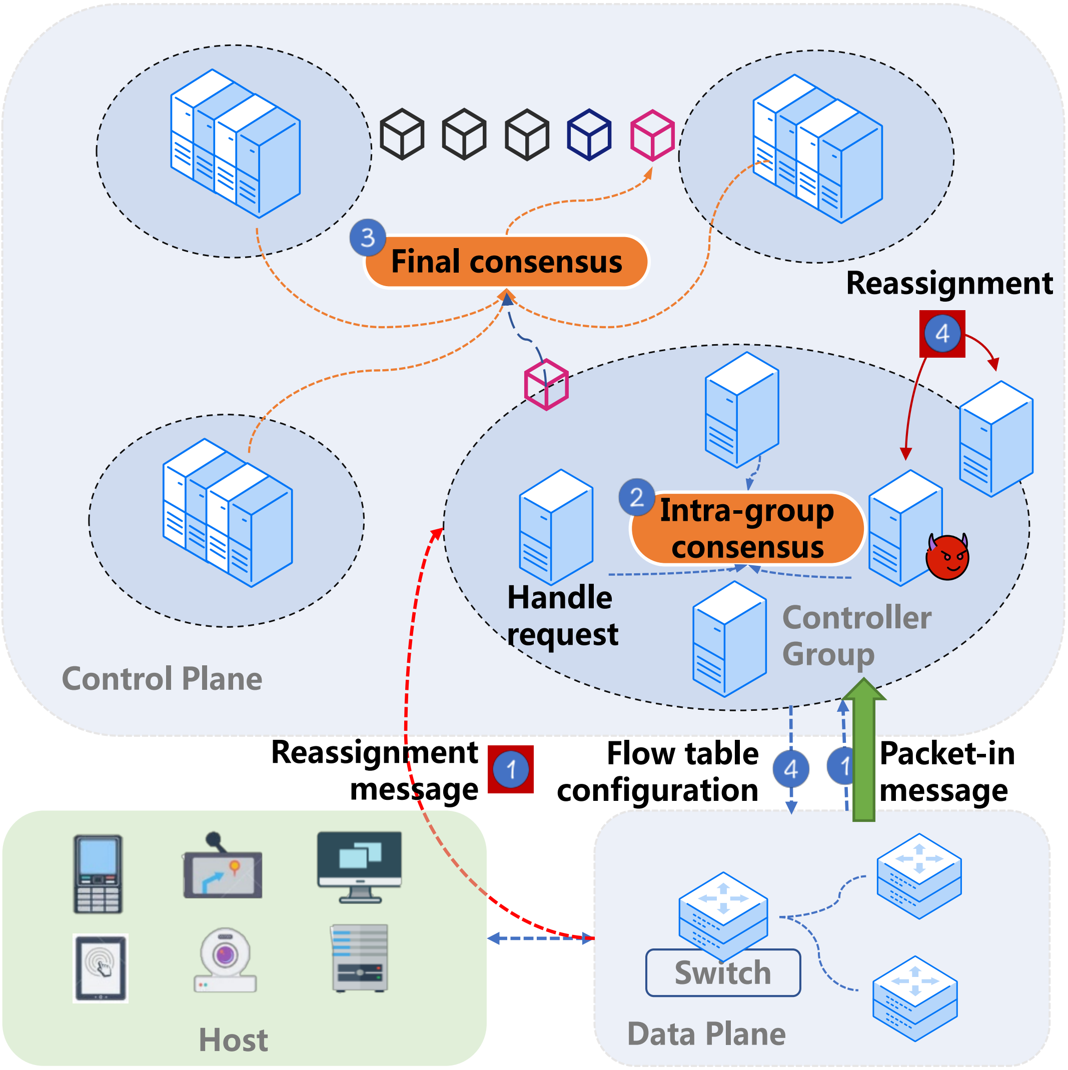
Workflow of Curb



Packet-in request

- **Step 0:** A user host sends a packet to the network so that it can be forwarded to its target host.
- **Step 1:** A switch sends a PKT-IN message to its assigned controller group to obtain forwarding rules.
- **Step 2:** The group members figure out forwarding rules and carry out the ***intra-group consensus*** process to reach consensus on the rules. After that they send blocks to the final committee.
- **Step 3:** The final committee takes charge of the ***final consensus*** process, where committee members reach consensus on blocks from multiple groups. After that the members broadcast blocks to every controller.
- **Step 4:** Controllers reply to switches with forwarding rules. Switches follow the forwarding rules to transmit packets if the rules are valid.

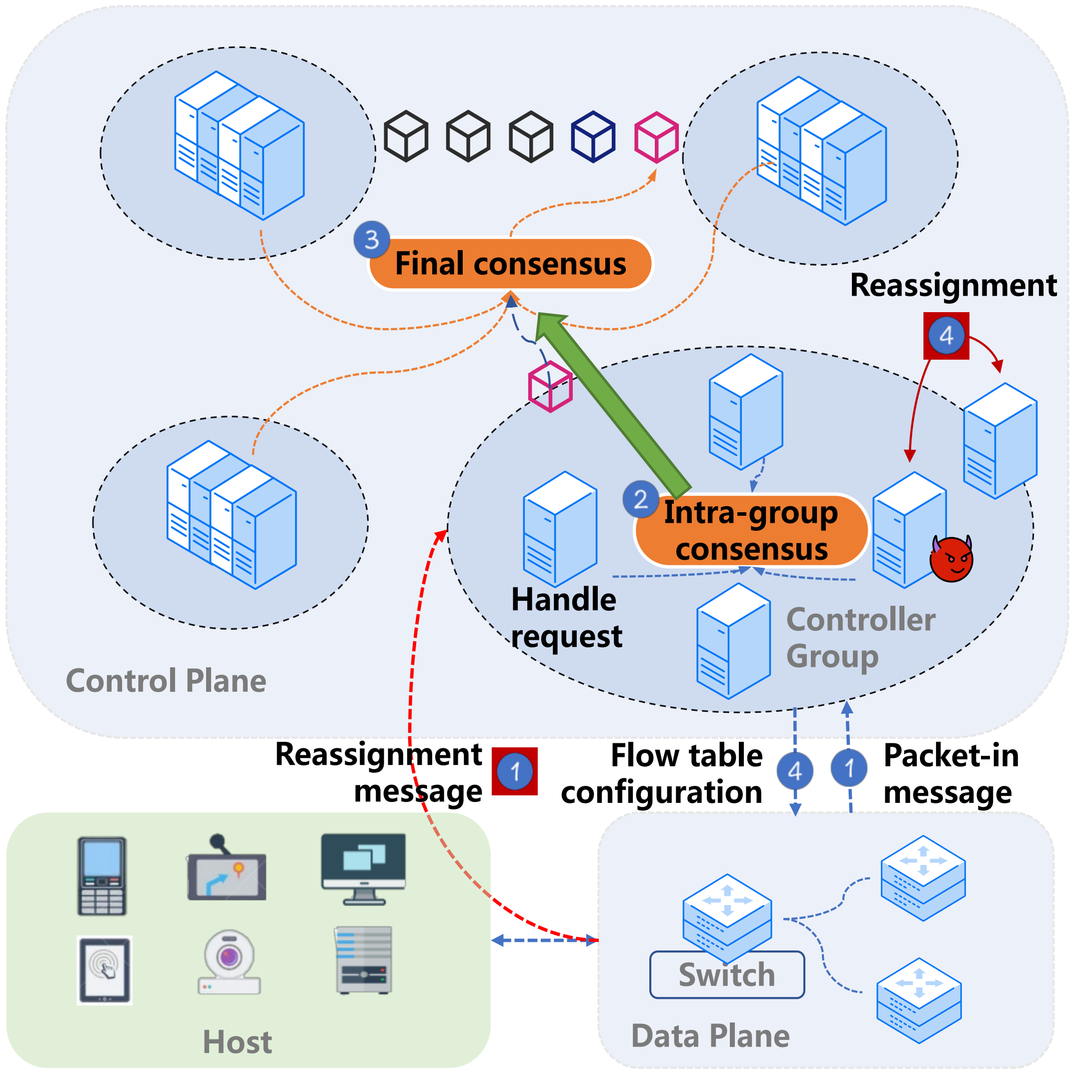
Workflow of Curb



Reassignment request

- **Step 1:** If a switch detects invalid replies, it will report the byzantine controllers in a **RE-ASS** message and broadcast the message to its assigned controller group.

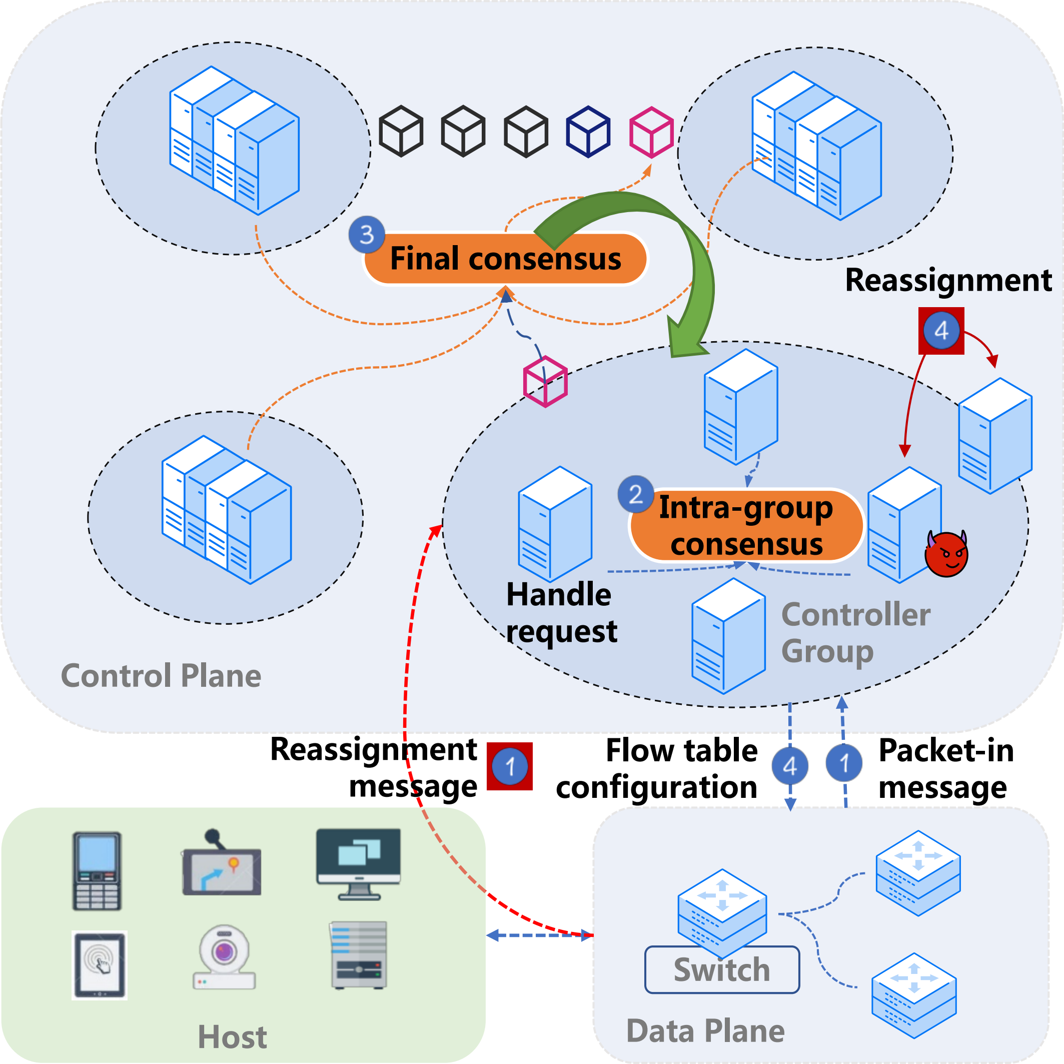
Workflow of Curb



Reassignment request

- **Step 1:** If a switch detects invalid replies, it will report the byzantine controllers in a **RE-ASS** message and broadcast the message to its assigned controller group.
- **Step 2:** The group members figure out a reassignment scheme and carry out the ***intra-group consensus*** process to reach consensus on the scheme. After that they send blocks to the final committee.

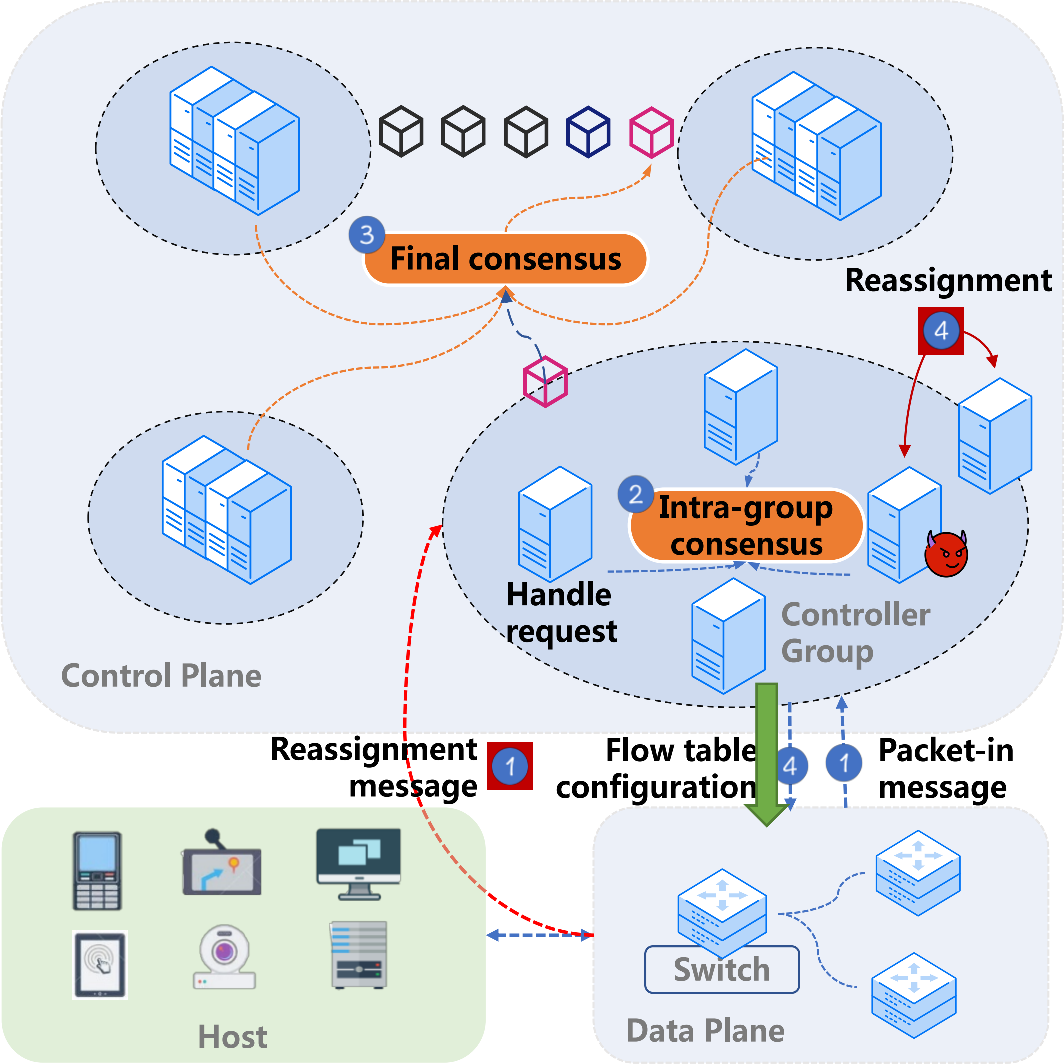
Workflow of Curb



Reassignment request

- **Step 1:** If a switch detects invalid replies, it will report the byzantine controllers in a **RE-ASS** message and broadcast the message to its assigned controller group.
- **Step 2:** The group members figure out a reassignment scheme and carry out the ***intra-group consensus*** process to reach consensus on the scheme. After that they send blocks to the final committee.
- **Step 3:** The final committee takes charge of the ***final consensus*** process, where committee members reach consensus on blocks from multiple groups. After that the members broadcast blocks to every controller.

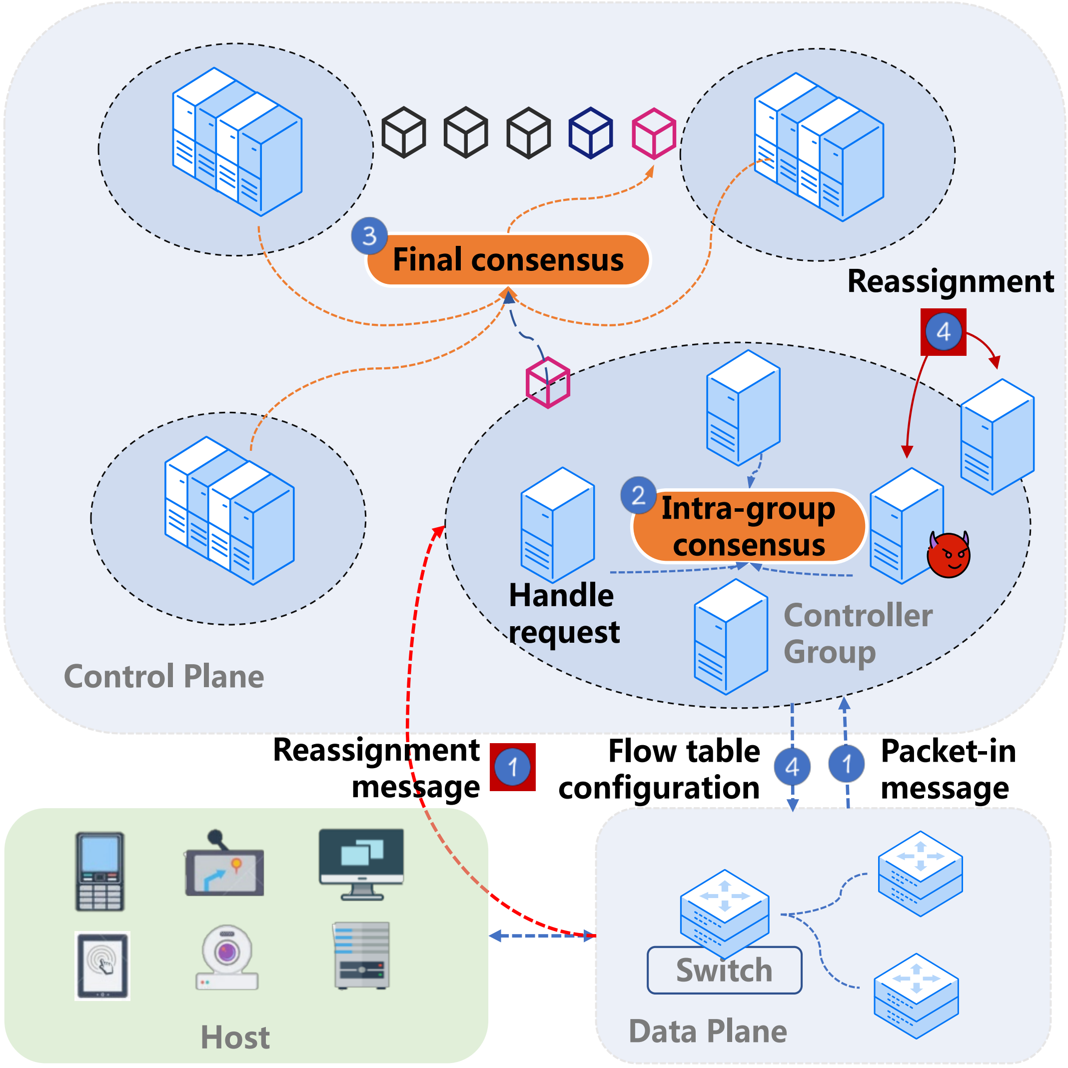
Workflow of Curb



Reassignment request

- **Step 1:** If a switch detects invalid replies, it will report the byzantine controllers in a **RE-ASS** message and broadcast the message to its assigned controller group.
- **Step 2:** The group members figure out a reassignment scheme and carry out the ***intra-group consensus*** process to reach consensus on the scheme. After that they send blocks to the final committee.
- **Step 3:** The final committee takes charge of the ***final consensus*** process, where committee members reach consensus on blocks from multiple groups. After that the members broadcast blocks to every controller.
- **Step 4:** Controllers reply to switches with the reassignment scheme. If the scheme is valid, controllers and switches will reconfigure the controller-to-controller (C2C) and controller-to-switch (C2S) links.

Analysis



Message complexity

- The number of groups: k
- The average group size: c
- The number of controllers: N

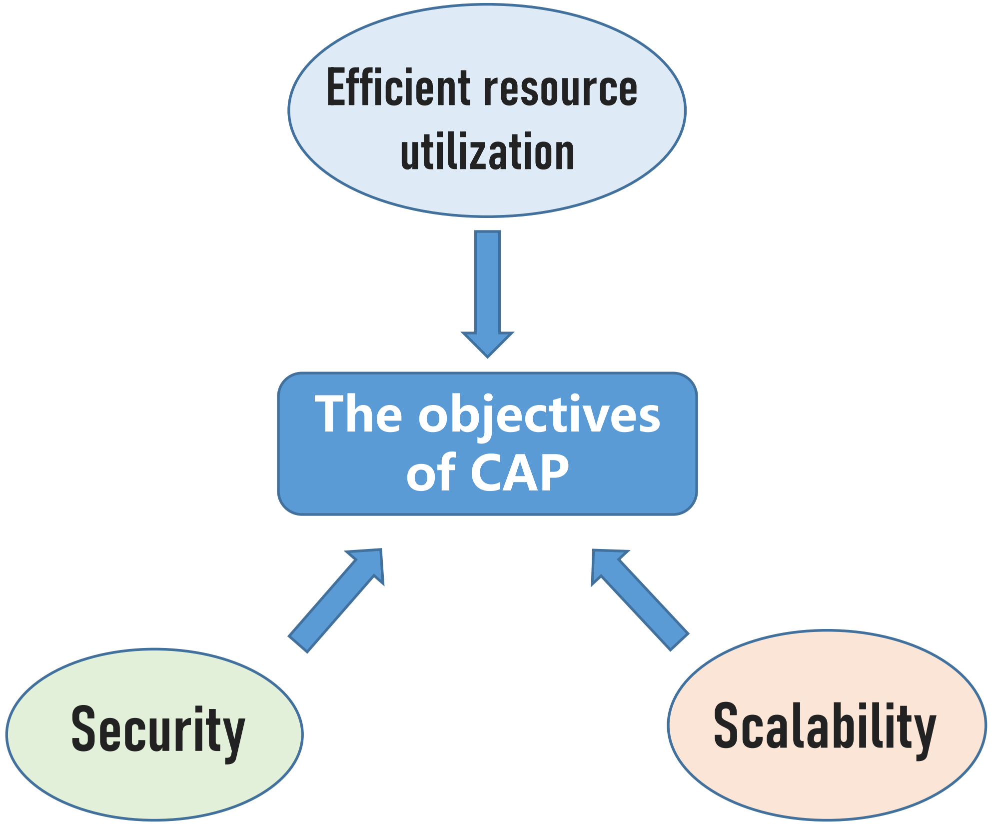
$$\boldsymbol{O(N) = O(k \times c)}$$

✓ **Step 1:** $O(N)$
✓ **Step 2:** $O(kc^2) + O(Nc)$
✓ **Step 3:** $O(c^2) + O(cN)$
✓ **Step 4:** $O(N)$

The message complexity of Curb is O(N), where N is the number of SDN controllers.

Analysis

The controller assignment problem (CAP)



Analysis

The controller assignment problem (CAP)

$$[O1] \quad \min \sum_{j \in C} x_j$$

$$\frac{1}{N} \sum_{i \in S} A_{ij} \leq x_j \leq 1 \quad \forall j \in C$$

Minimizing the number of used controllers

Maximizing the utilization of each controller

Efficient resource utilization

$$[C1.1] \quad \sum_{i \in S} A_{ij} Q_i \leq C_j \quad \forall j \in C$$

$$[C1.2] \quad \sum_{j \in C} A_{ij} \geq B_i \quad \forall i \in S$$

Security: the size of each controller group should be more than 3f+1, where f is the maximum number of faulty nodes in a group.

$$[C1.3] \quad A_{ij} d_{ij} \leq D_{c,s} \quad \forall i \in S, \forall j \in C$$

$$[C1.4] \quad A_{ij} A_{ij'} d_{ij'} \leq D_{c,c} \quad j \neq j', \forall j, j' \in C, \forall i \in S$$

Scalability: reducing the C2C and C2S link delay in each group.

Analysis

The controller reassignment problem

[$C2.5$] $x_j = 0 \quad \forall j \in C_{byz}$ — Removing byzantine nodes

[$C2.6$] $A_{ij} = 1 \quad \forall (i,j) \in LEADER$ — Fixing honest leader nodes

[$O3$] $$\text{LCR : min} \left\{ \sum_{j \in C} x_j + \sum_{j \in C \wedge i \in S} |A_{ij} - a_{ij}| \right\}$$

- Minimizing the number of used controllers
- Minimizing the number of changed links

[$O2$] $$\text{TCR : min} \sum_{j \in C} x_j$$

Minimizing the number of used controllers

Evaluation

Experiment configuration

- ✓ Mininet + Ryu
- ✓ Internet2 network (16 controllers, 34 switches)
- ✓ Gurobi optimizer

Internet2 topology

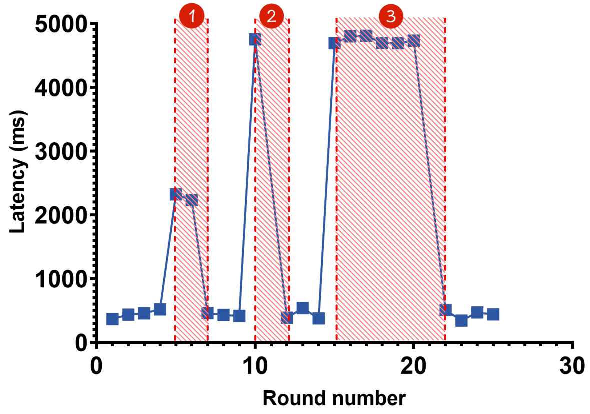
Tests on:

- ✓ Curb’s capability of defending against byzantine nodes;
- ✓ The performance of handling the packet-in requests;
- ✓ The performance of two types of optimization programming solvers for controller reassignment;
- ✓ The performance of handling the reassignment requests.

Evaluation

Byzantine resilience test

- **Experiment ❶: one byzantine node** does not respond to any request starting from the 5th round, and is removed in the 6th round.
- **Experiment ❷: three byzantine nodes** do not respond to any request starting from the 10th round, and are removed in the 11th round.
- **Experiment ❸: three lazy nodes** respond to requests slowly starting from the 15th round, and are removed in the 21th round.



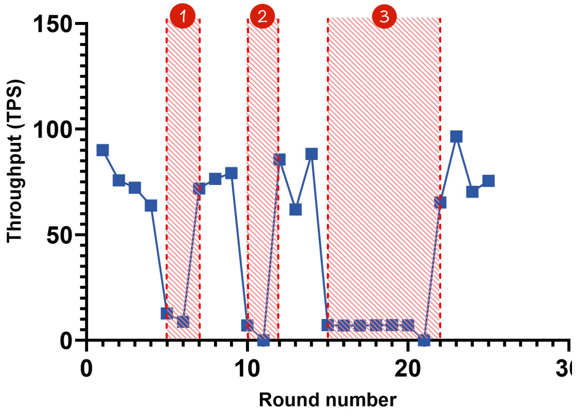
(a) Latency *vs*. round

Remarks

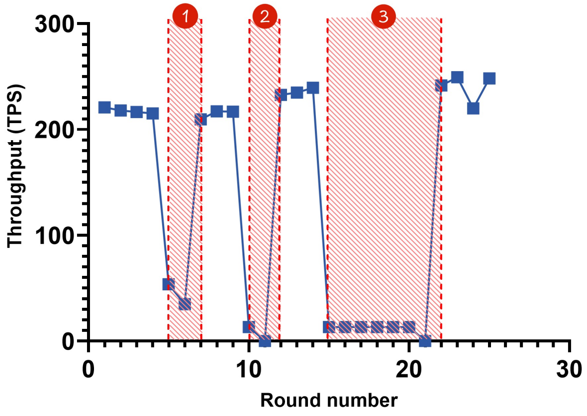
✓ Fault-tolerant resilience;

✓ Latency: 460.24 ms and throughput: 71.90 TPS;

✓ The parallel processing mode significantly improves the throughput.



(b) Throughput vs. round (non-parallel)

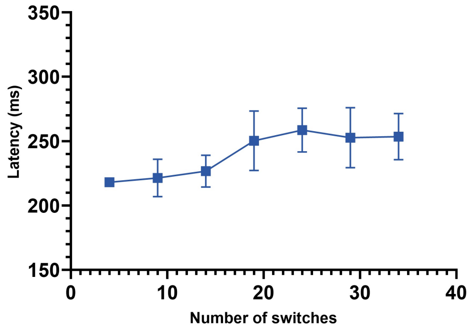


(c) Throughput vs. round (parallel)

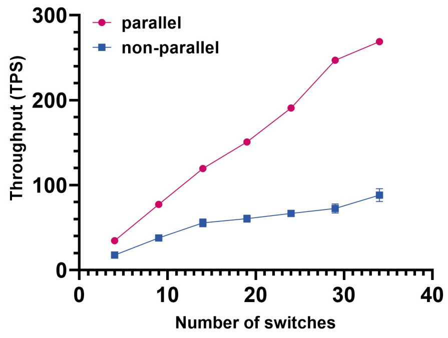
Evaluation

Performance of handling the packet-in requests

- How is the performance impacted by the network scale?
  - The number of switches ↗
  - The value of f ↗



(a) Latency vs. the number of switches



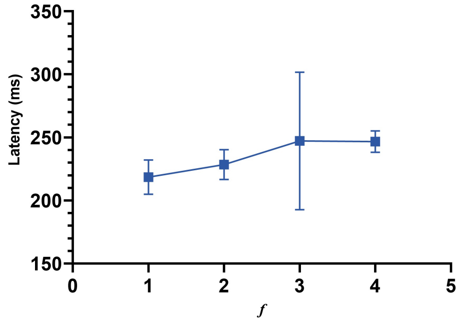
(b) Throughput vs. the number of switches

Remarks

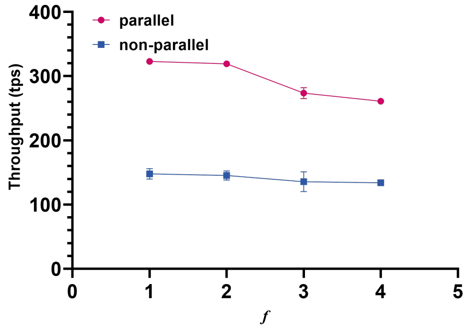
✓ The latency slightly increases with the number of switches and the value of f.

✓ The throughput linearly increases with the number of switches.

✓ The throughput slightly decreases with the value of f.



(c) Latency *vs.* f



(d) Throughput *vs.* f

Evaluation

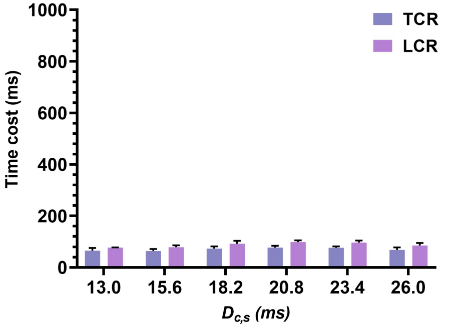
Performance of the optimization programming

Time cost vs. $D_{c,s}$

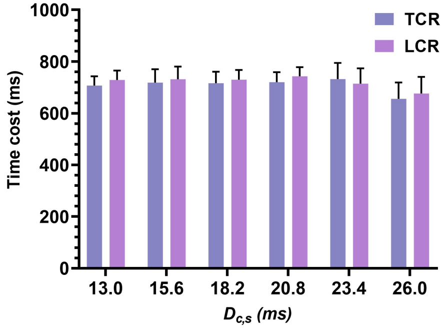
- Compare TCR and LCR with varying $D_{c,s}$ under different combinations of the following constraints.

[C2.4] $A_{ij}A_{ij'}d_{ij'} \leq D_{c,c}$ (the upper bound of C2C link delay)

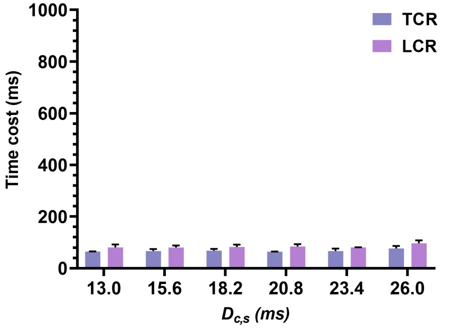
[C2.6] $A_{ij} = 1 \quad \forall (i,j) \in LEADER$ (fixing leader nodes)



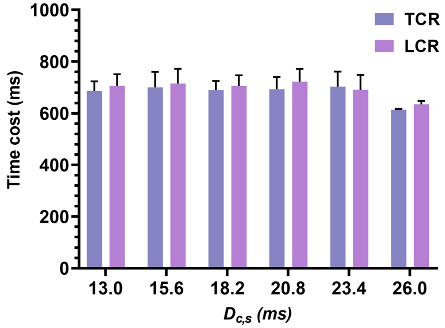
(a) With the leader constraint



(b) With the $D_{c,c}$ constraint



(c) Without the leader and $D_{c,c}$ constraints



(d) With the leader and $D_{c,c}$ constraints

Remarks

Nonlinearity
- LCR costs a little more time than TCR.
- The $D_{c,c}$ constraint leads to significant time overheads.

Evaluation

Performance of the optimization programming

The number of used controllers vs. $D_{c,s}$

- Compare TCR and LCR with varying $D_{c,s}$ under different combinations of the following constraints.

[C2.4] $A_{ij}A_{ij'}d_{ij'} \le D_{c,c}$ (the upper bound of C2C link delay)

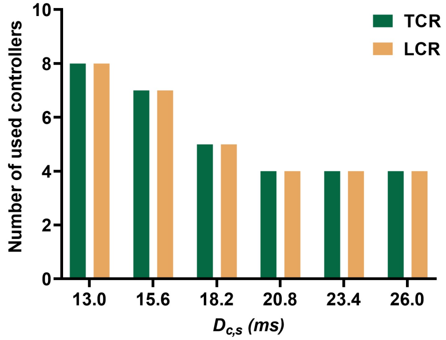
[C2.6] $A_{ij} = 1 \quad \forall (i,j) \in LEADER$ (fixing leader nodes)

Remarks

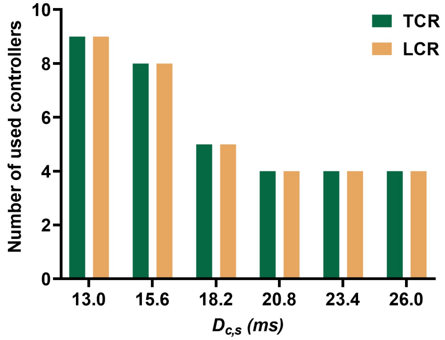
✓ The TCR and LCR methods output the same number of controllers being used.

✓ Less controllers is used if $D_{c,s}$ is higher.

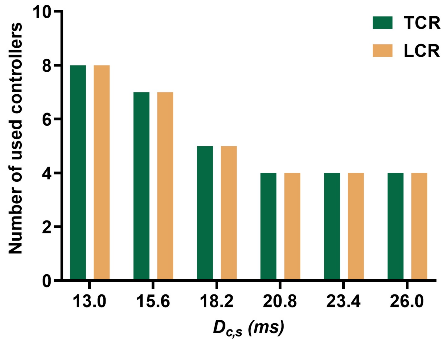
✓Adding the $D_{c,c}$ constraint can result in more controllers enrolled.



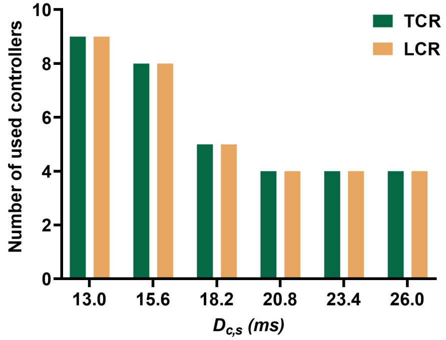
(a) With the leader constraint



(b) With the $D_{c,c}$ constraint



(c) Without the leader and $D_{c,c}$ constraints



(d) With the leader and $D_{c,c}$ constraints

Evaluation

Performance of the optimization programming

The percentage of dynamic links (PDL) vs. $D_{c,s}$

- Compare TCR and LCR with varying $D_{c,s}$ under different combinations of the following constraints.

$$[C2.4] \quad A_{ij}A_{ij'}d_{ij'} \le D_{c,c} \quad \text{(the upper bound of C2C link delay)}$$

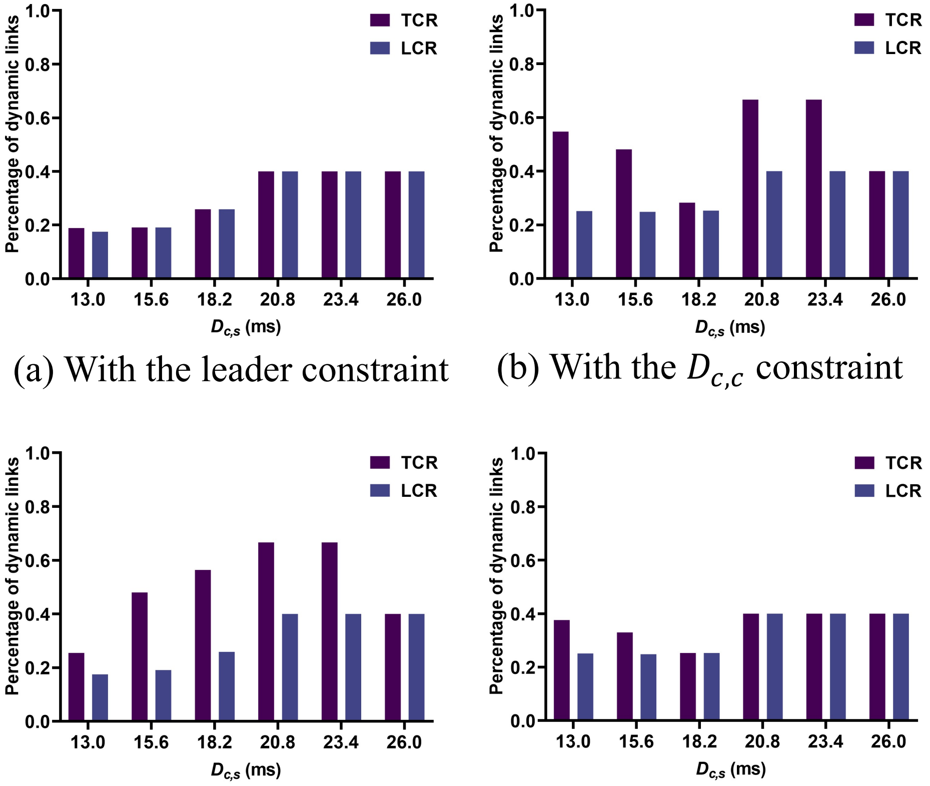
$$[C2.6] \quad A_{ij} = 1 \quad \forall (i,j) \in LEADER \quad \text{(fixing leader nodes)}$$

Remarks

✓ Less links are changed with a lower $D_{c,s}$.

✓ LCR has better performance of PDL than TCR.

✓ Bringing the leader constraint can result in less PDL.



(a) With the leader constraint

(b) With the $D_{c,c}$ constraint

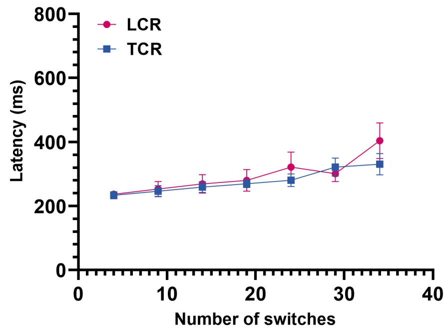
(c) Without the leader and $D_{c,c}$ constraints

(d) With the leader and $D_{c,c}$ constraints

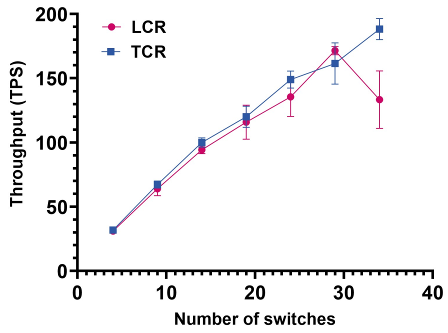
Evaluation

Performance of handling the reassignment requests

- How is the performance impacted by the network scale, when the system handles a large number of reassignment requests?
  - The number of switches ↗
  - The value of f ↗



(a) Latency vs. the number of switches



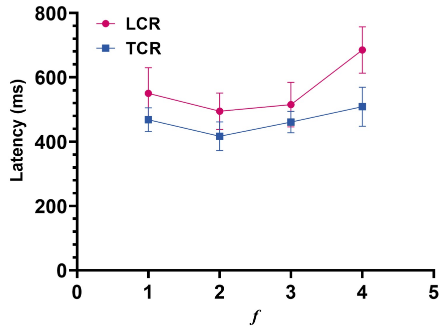
(b) Throughput vs. the number of switches

Remarks

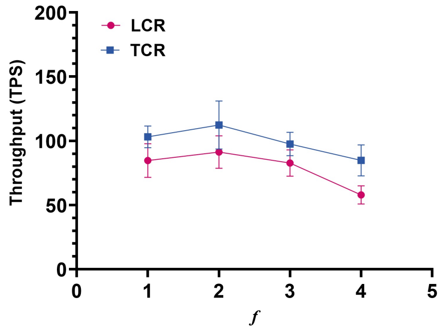
✓ The latency with TCR and LCR solvers is very close with the increasing number of switches.

✓ The extra time cost of LCR compared to TCR become more explicit with a higher f.

✓ The throughput still linearly increases with the number of switches and slightly decreases with the value of f.



(c) Latency *vs.* f



(d) Throughput *vs.* f

Conclusion

- ✓ We present Curb, a novel SDN control plane scheme that seamlessly integrates blockchain and BFT consensus into a group-based control plane, addressing security and scalability concerns of the state-of-the-arts.
- ✓ Curb supports trusted flow rule updates and adaptive controller reassignment.
- ✓ Curb uses a group-based technique to realize a scalable network where the message complexity of each round is upper bounded by $O(N)$.

Q&A

Thank you for your listening!