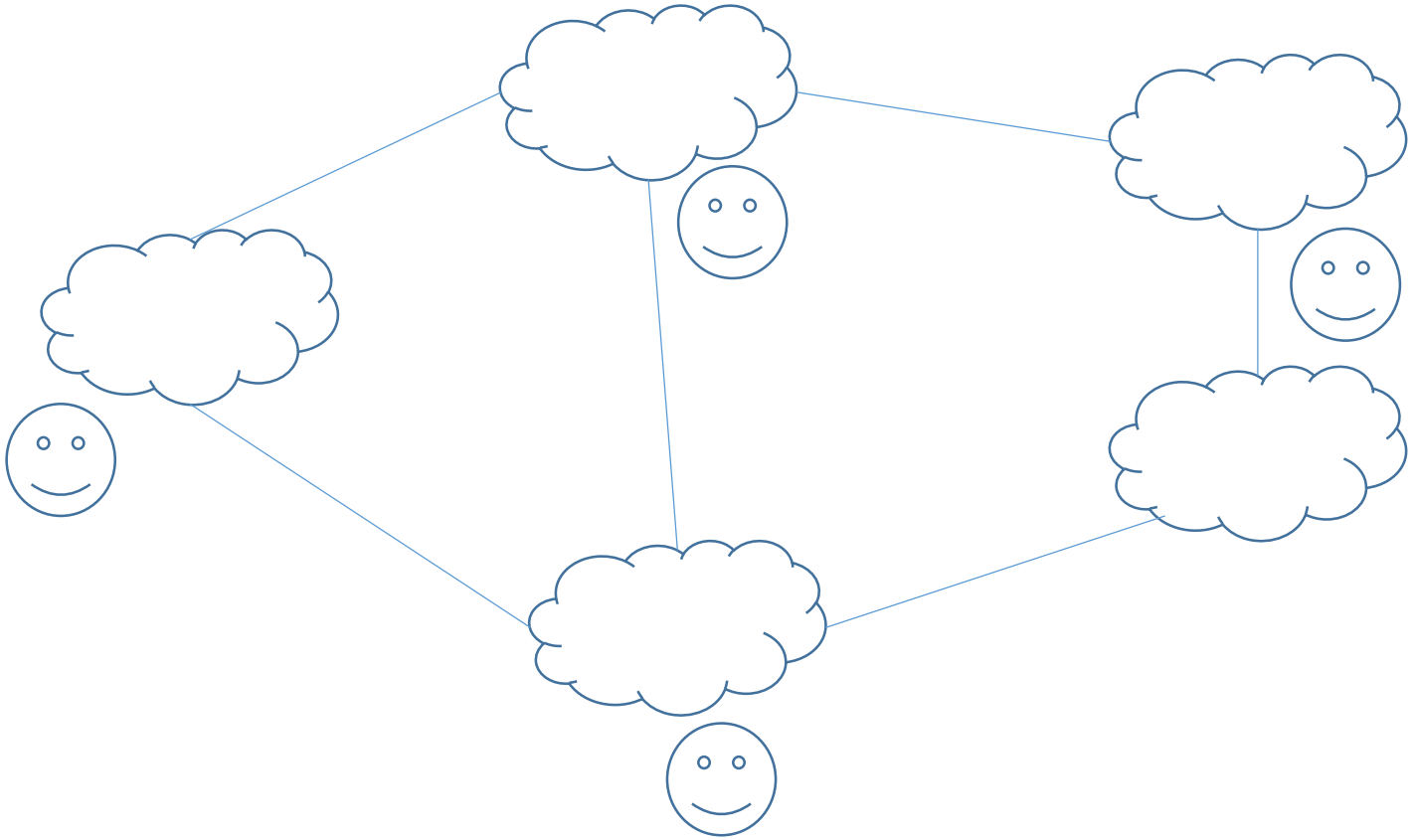
Firenet – PhD semester project

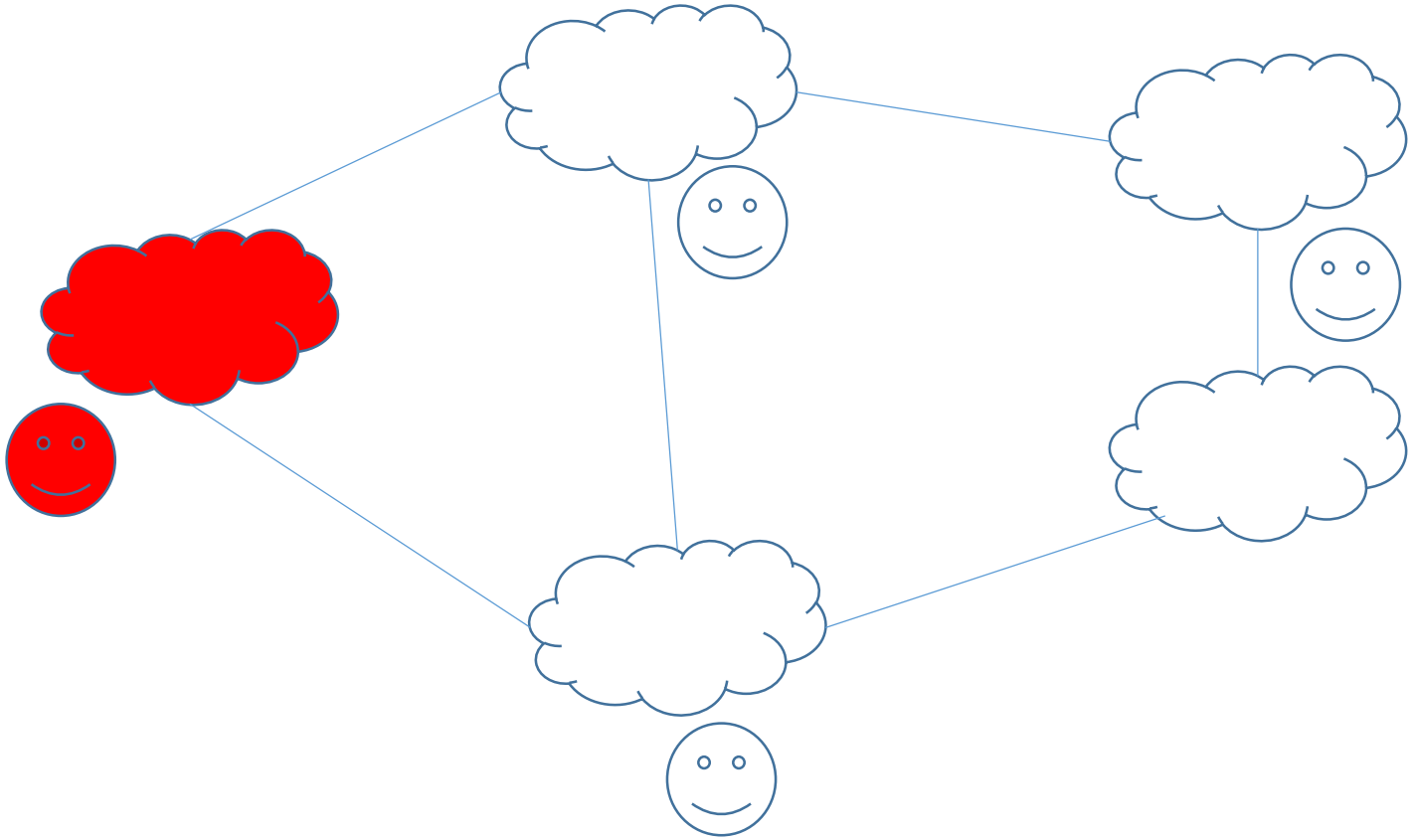
Jingyue Zhao

Supervisors: Prof. Bryan Ford, Prof. Katerina Argyraki

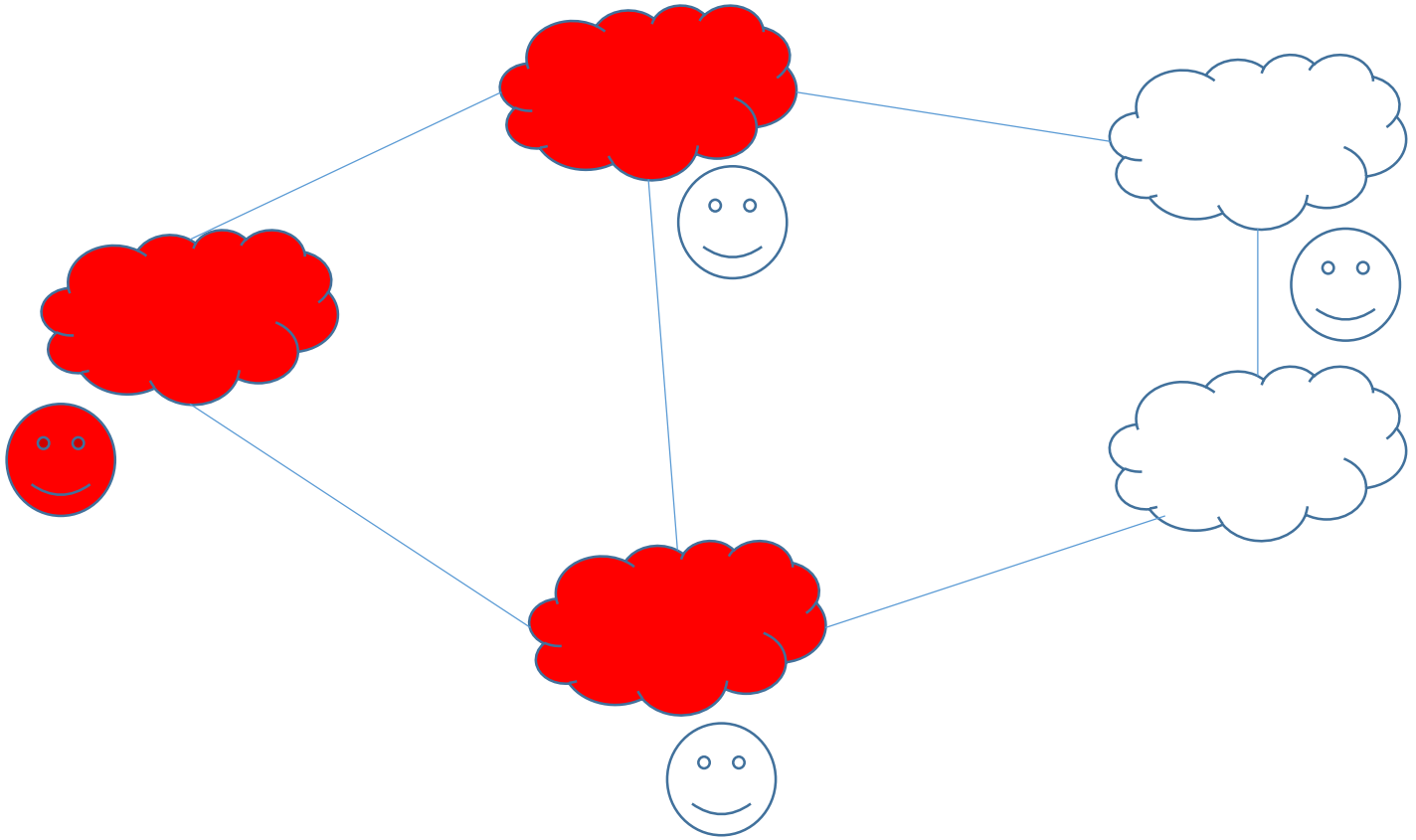
Network Management



Network Management



Network Management

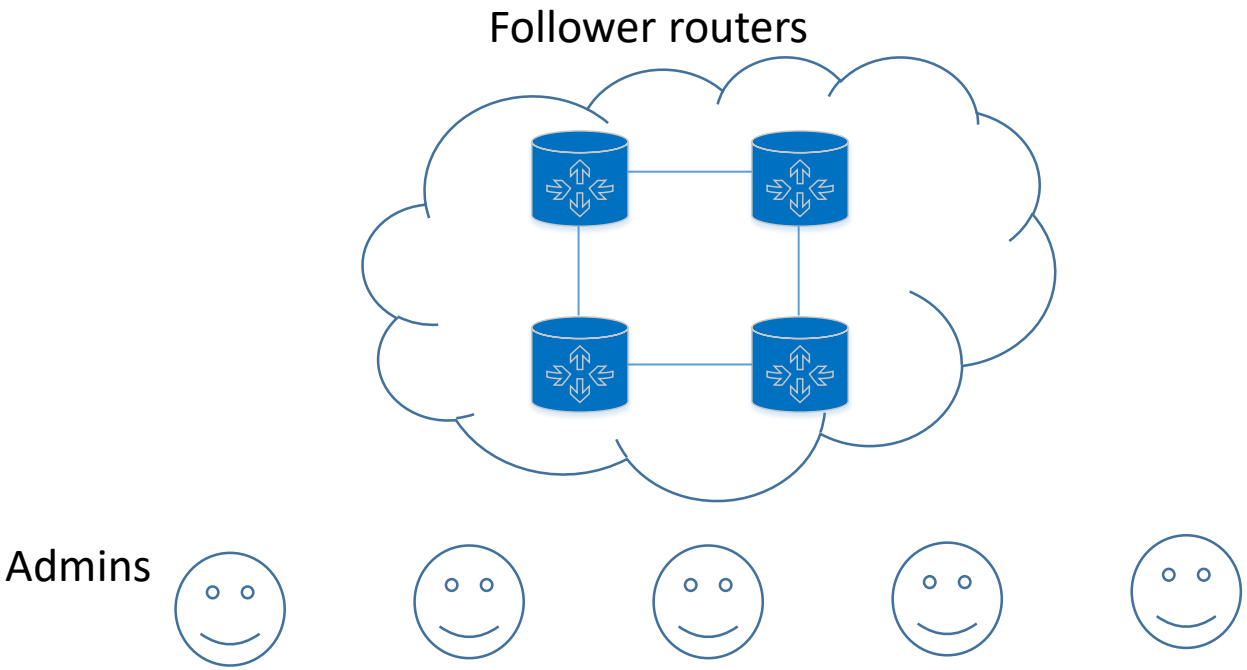


Motivation

- Long-term goal: a transparent and secure decentralized network management scheme for large-scale networks.
- Decisions of each administrator →direct or indirect impacts on other parts of network.
- Admins can be compromised → disaster of the entire network.

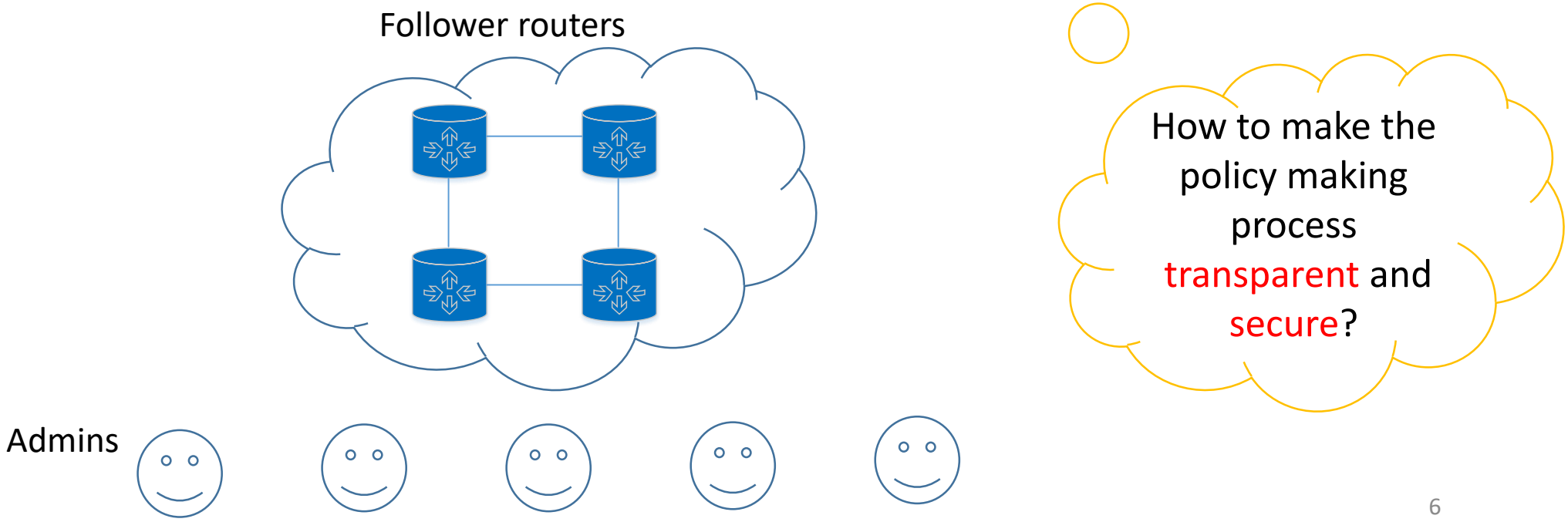
System Model

- A single small group of administrators make network policies together.
- Follower routers correspond to SDN controllers which deploy the network policies.
- Each network policy needs to be checked and approved by a threshold of admins to avoid careless or malicious actions.



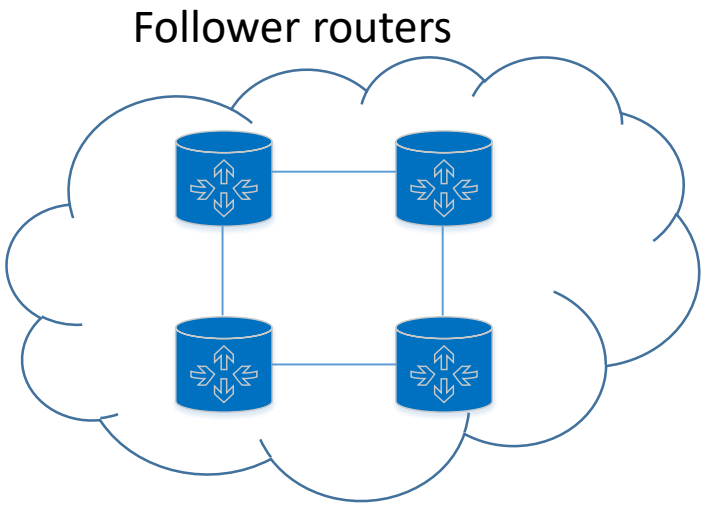
System Model

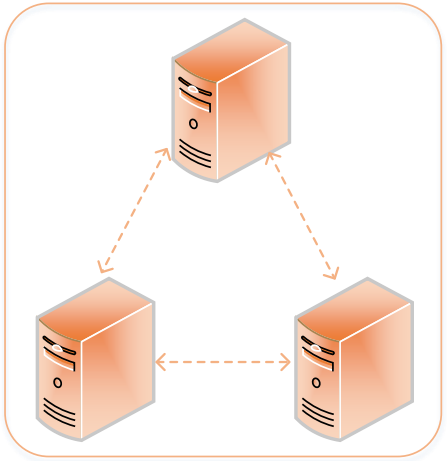
- A single small group of administrators make network policies together.
- Follower routers correspond to SDN controllers which deploy the network policies.
- Each network policy needs to be checked and approved by a threshold of admins to avoid careless or malicious actions.



System Model

- A single small group of administrators make network policies together.
- Follower routers correspond to SDN controllers which deploy the network policies.
- Each network policy needs to be checked and approved by a threshold of admins to avoid careless or malicious actions.

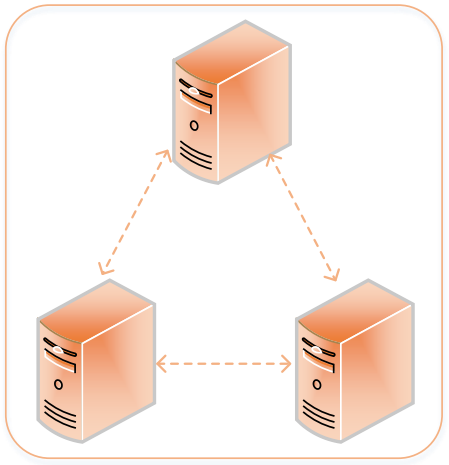




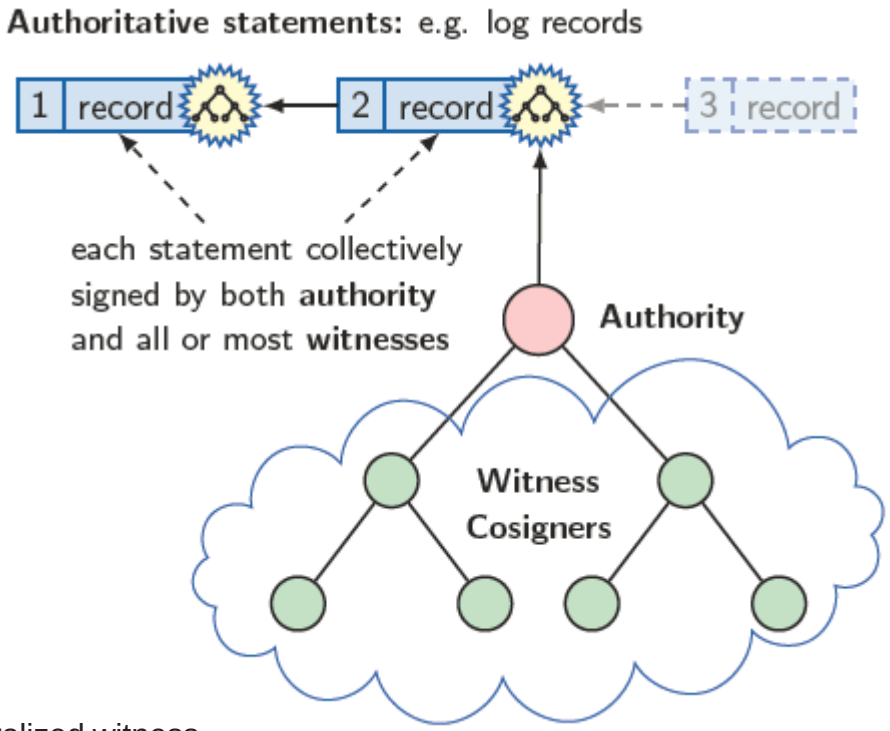
Cothority

Cothority

- Collective authority: a set of witness servers called conodes that collectively execute decentralized protocols
- Provide services: collective signing (CoSi [1]), Byzantine agreement, or generation of public-randomness



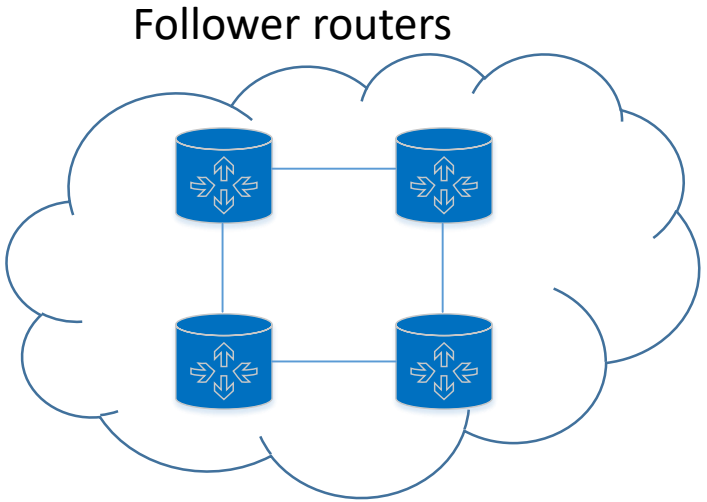
Cothority

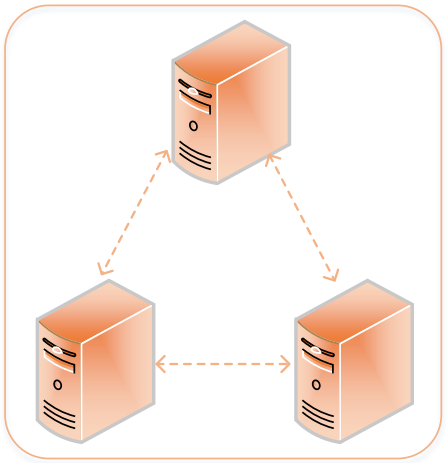


[1] Syta E, Tamas I, Visher D, et al. Keeping authorities" honest or bust" with decentralized witness cosigning[C]//Security and Privacy (SP), 2016 IEEE Symposium on. Ieee, 2016: 526-545.

Threat Model

- Network administrators: make and approve network policies
  - Malicious: propose or approve bad policies; only a threshold of admins can be compromised by an attacker
- Cothority (witness servers): check and track admins' policy making process
  - Honest
- Follower routers: pull and deploy the network policies periodically
  - Honest





Design of Firenet

Step 1: admins' approval


Key 1


Key 2

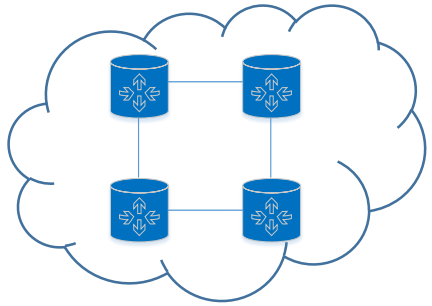

Key 3


Key 4

Admins

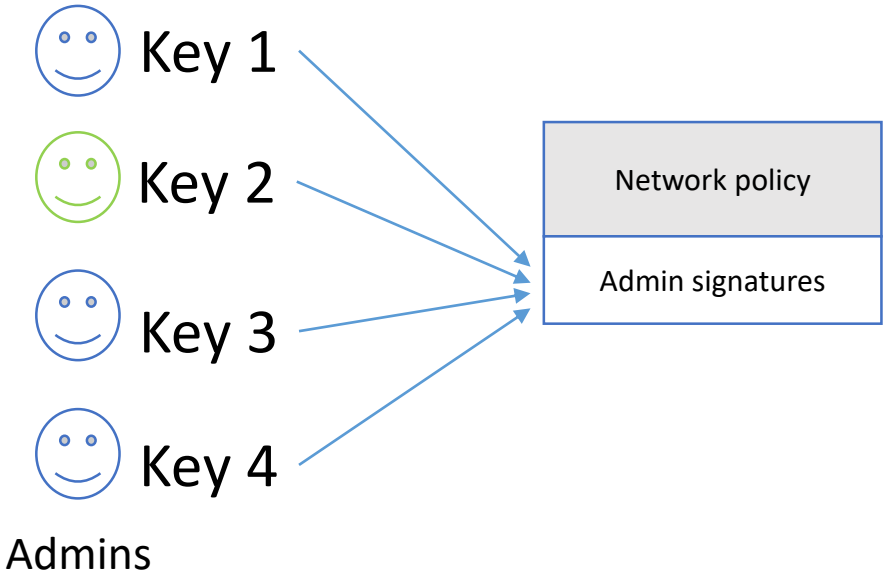
Network policy

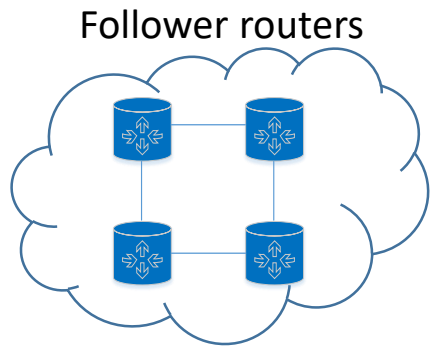
Follower routers



Design of Firenet

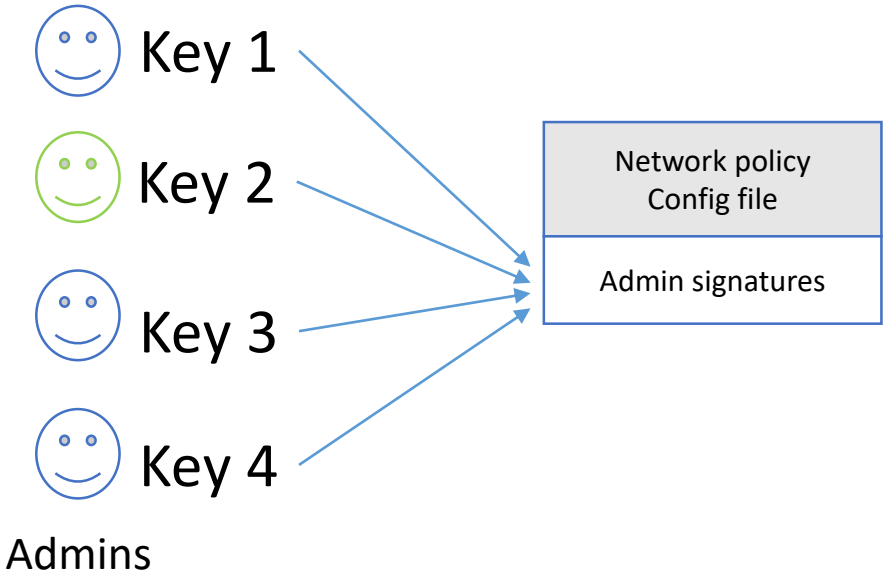
Step 1: admins' approval

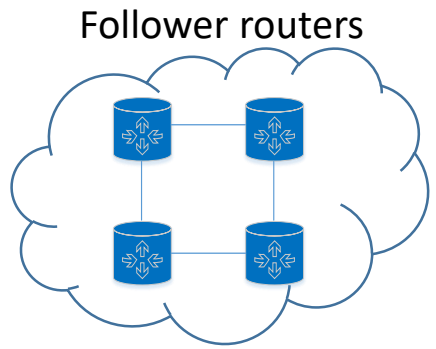




Design of Firenet

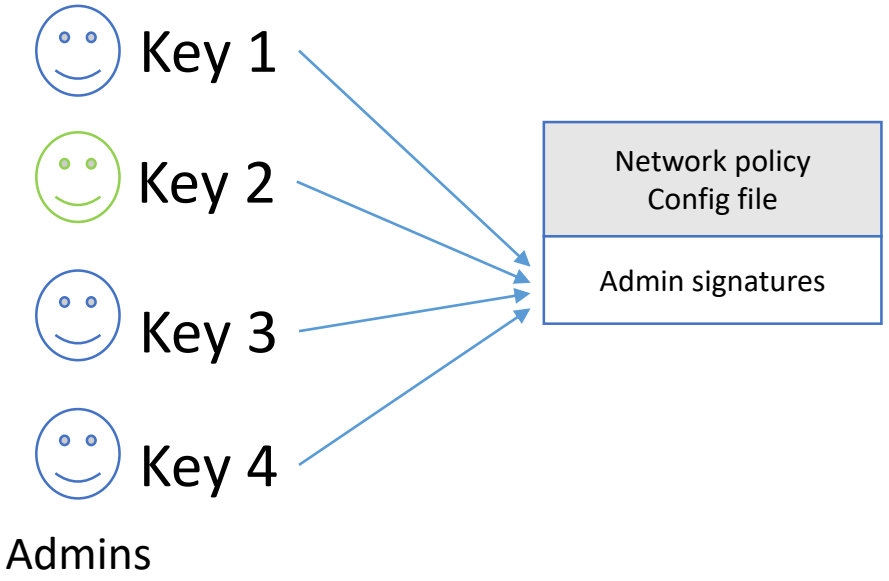
Step 1: admins' approval

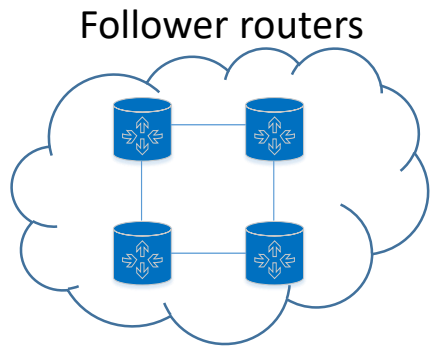




Design of Firenet

Step 1: admins' approval

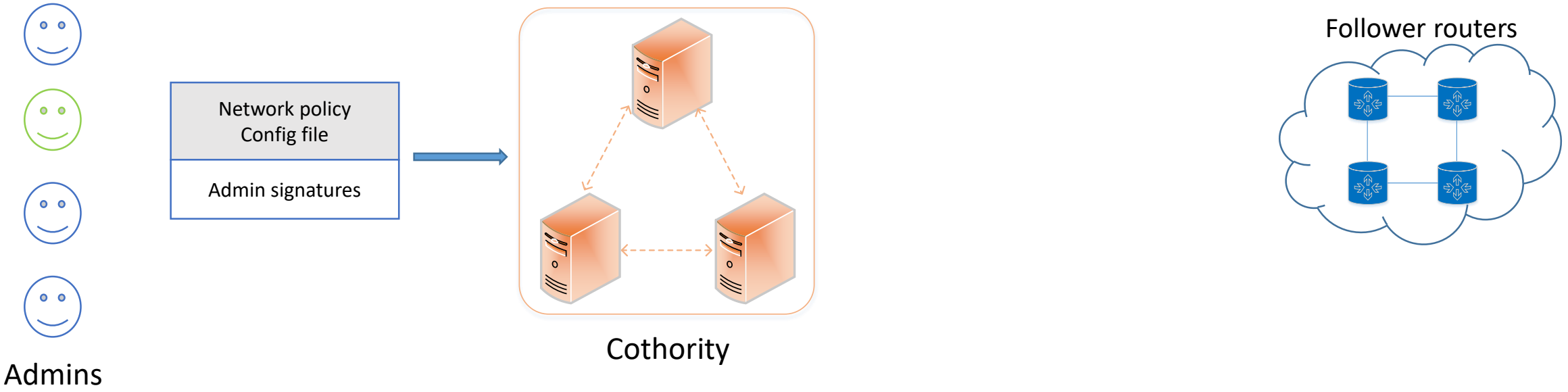




Verify admins' signatures & deploy the policy

Design of Firenet

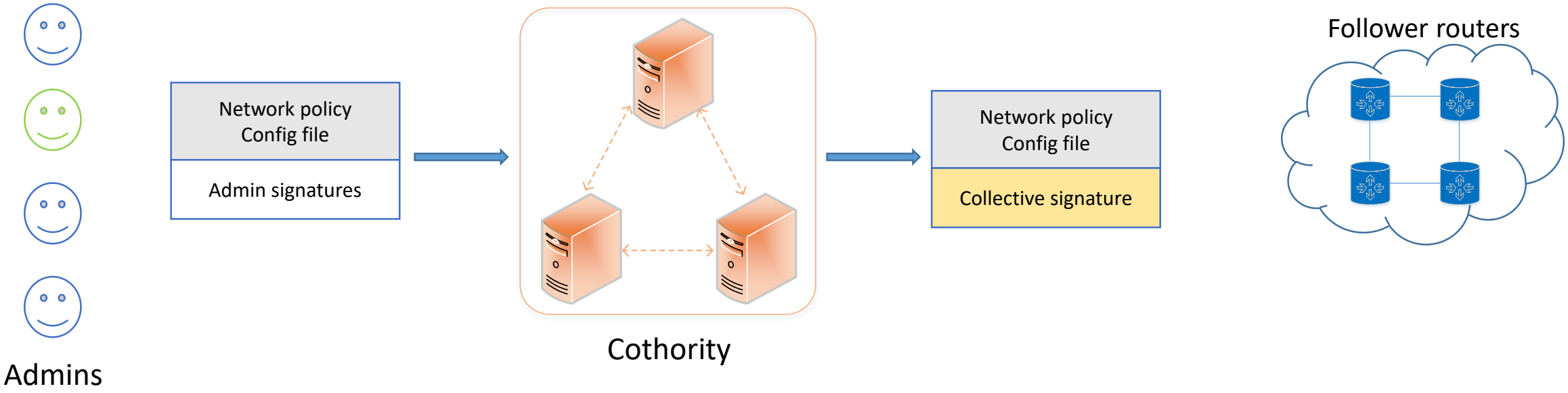
Step 2: cothority's approval check and collective signing



For now, the check is done by one server in the cothority, and we can design a protocol to distribute the workload.

Design of Firenet

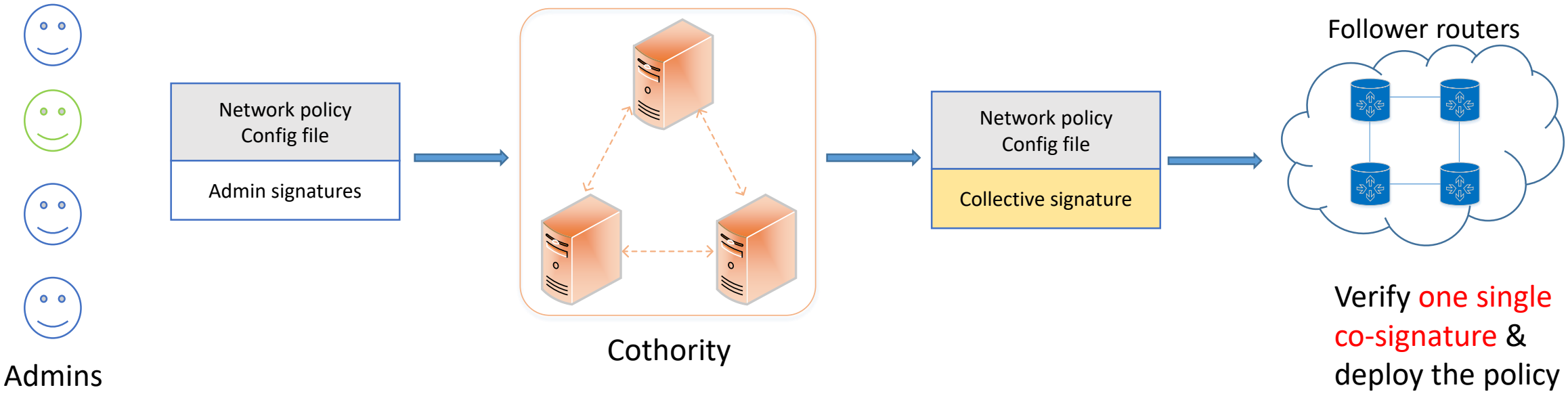
Step 2: cothority's approval check and collective signing (using CoSi [1])



[1] Syta E, Tamas I, Visher D, et al. Keeping authorities" honest or bust" with decentralized witness cosigning[C]//Security and Privacy (SP), 2016 IEEE Symposium on. Ieee, 2016: 526-545.

Design of Firenet

Step 2: cothority's approval check and collective signing



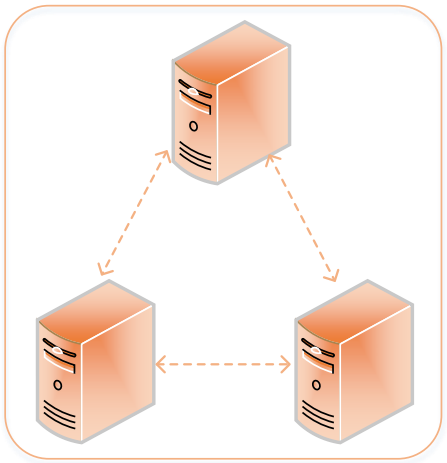
Design of Firenet

Step 3: cothority's appending the new policy to the chain (using Skipchain[2])

Admins

Network policy
Config file
Previous block hash

Admin signatures



Cothority

1 Block hash

Network policy
Config

Collective signature

Random ID

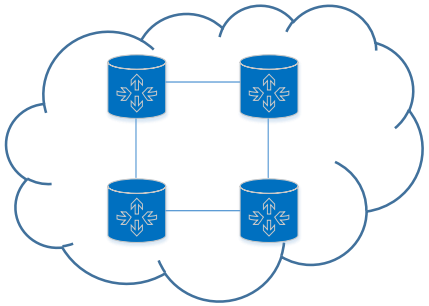
2 Block hash

Network policy
Config

Collective signature

Previous block hash

Follower routers

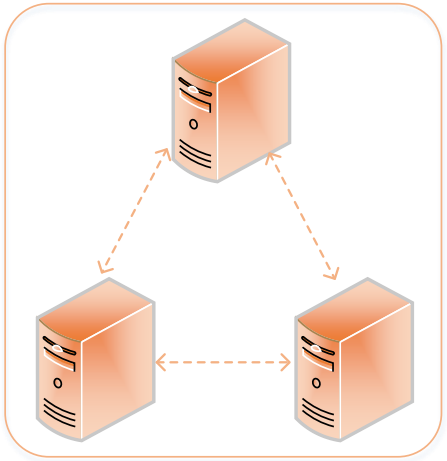


[2] Nikitin K, Kokoris-Kogias L, Jovanovic P, et al. CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds[J]. 2017.

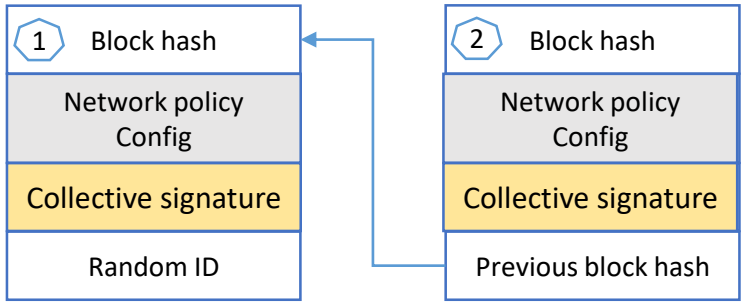
Design of Firenet

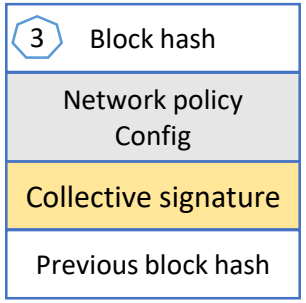
Step 3: cothority's appending the new policy to the chain

Admins

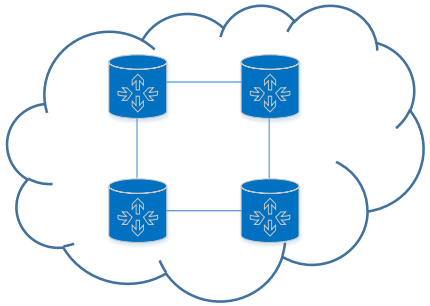


Cothority





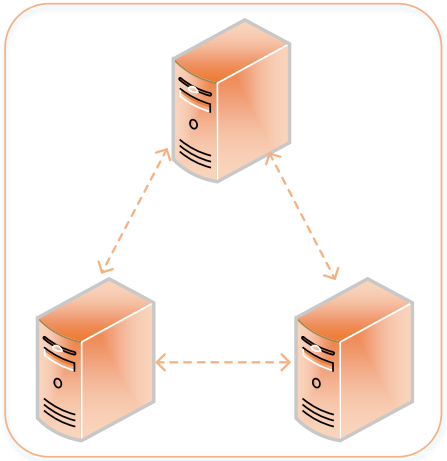
Follower routers



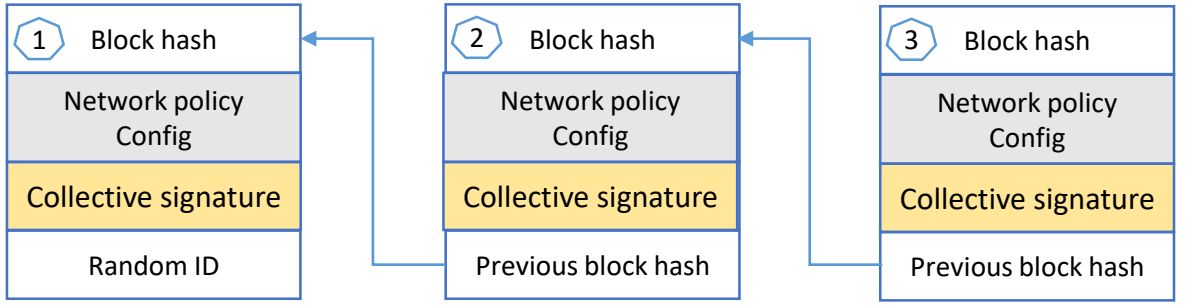
Design of Firenet

Step 3: cothority's appending the new policy to the chain

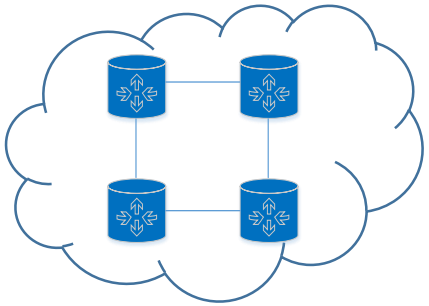
Admins



Cothority



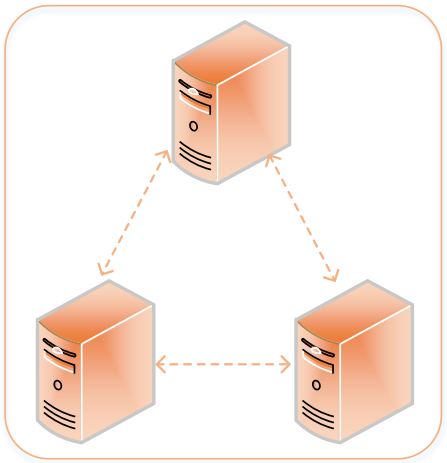
Follower routers



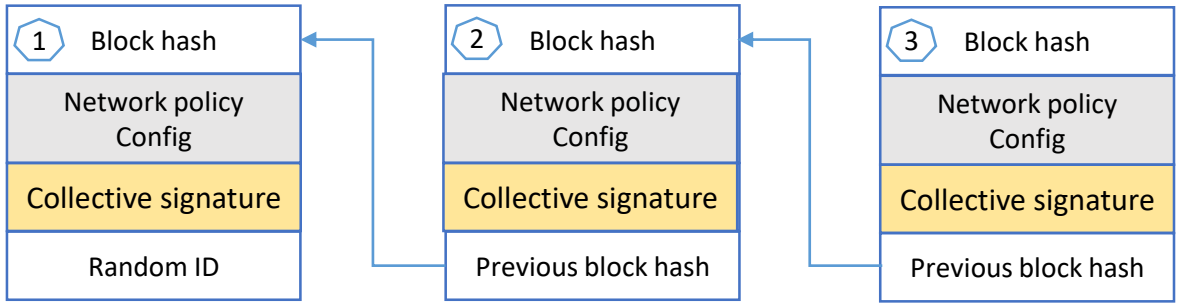
Design of Firenet

Step 3: cothority's appending the new policy to the chain

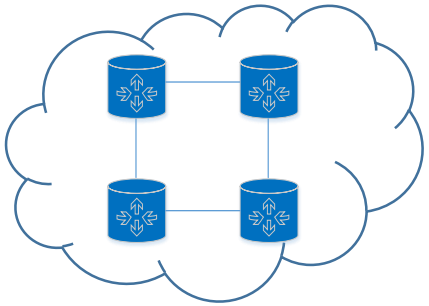
Admins



Cothority



Follower routers

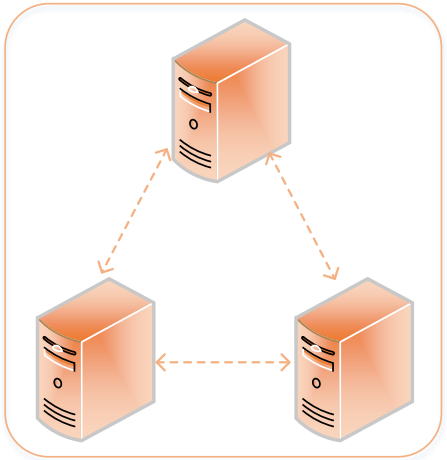


Firenet in 1 slide

Admins

Network policy Config file Previous block hash
Admin signatures

Step 1: Admins' policy proposal & approval



Cothority

1 Block hash
Network policy Config
Collective signature
Random ID

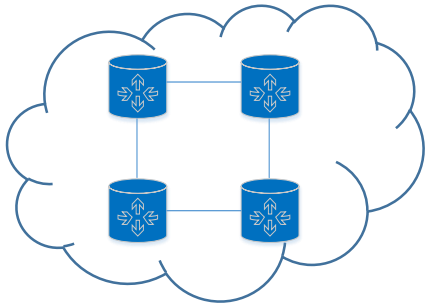
2 Block hash
Network policy Config
Collective signature
Previous block hash

3 Block hash
Network policy Config
Collective signature
Previous block hash

Step 2: Cothority's approval check & co-signature

Step 3: Cothority's appending new policy to the chain

Follower routers



Step 4: Follower routers' downloading, verifying & deploying the latest policy periodically

Network Policy Description Language

- Based on Linux iptables
- One network policy consists of several network rules
- One policy is self-sufficient
- JSON object

Network policy

Property	**Value**
Policy description	string
Number of network rules	int
An array of network rules	Network rule 1
	Network rule 2
	...
	Network rule n

Network rule

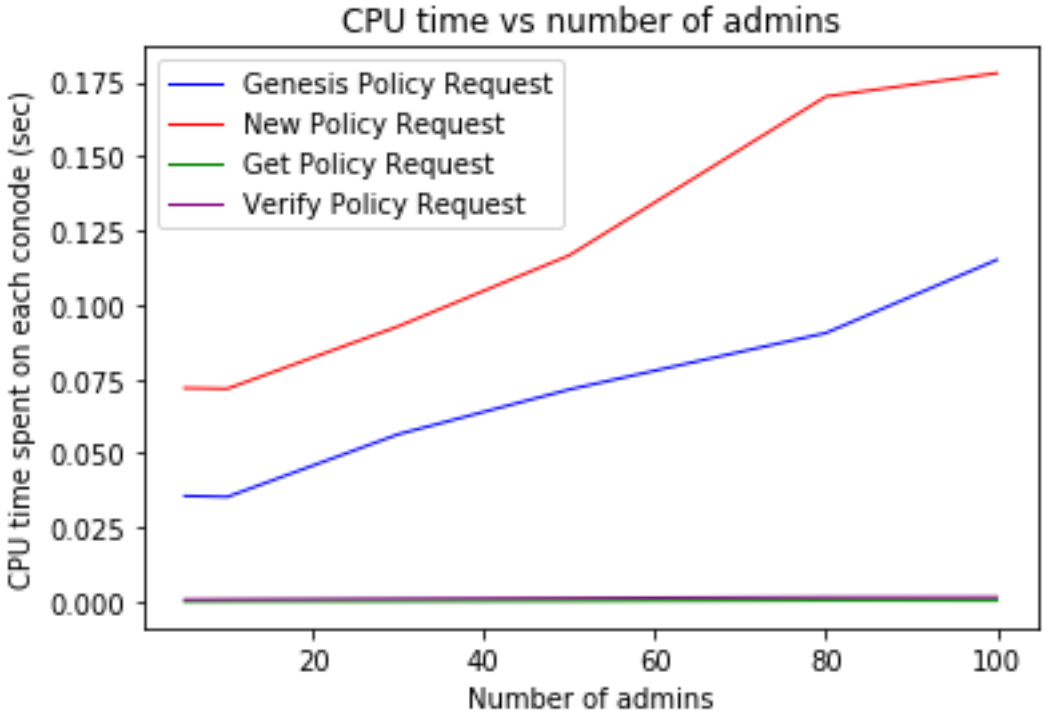
Property		**Value**
Matches	Chain	INPUT, OUTPUT, FORWARD
	Protocol	TCP, UDP, ICMP, ALL
	Source IP/network	x.x.x.x, x.x.x.x/x, ALL
	Source ports	Port number(s)
	Destination IP/network	x.x.x.x, x.x.x.x/x, ALL
	Destination ports	Port number(s)
Action		ACCEPT, DROP, REJECT

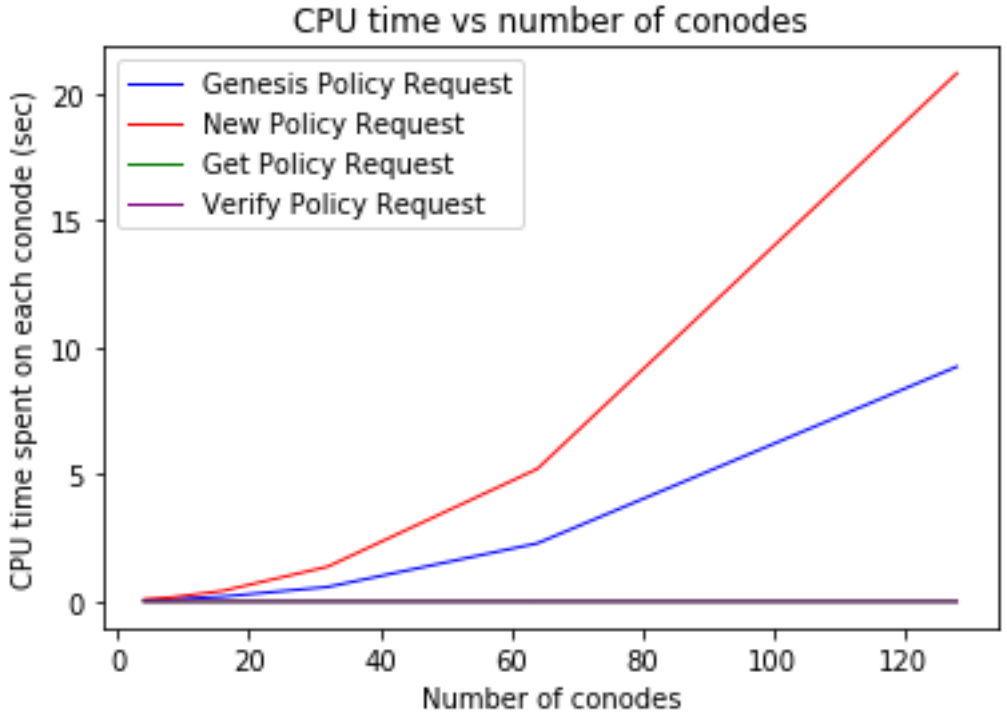
Implementation

- Firenet is implemented in Go
- Based on the Cothority framework, using CoSi and Skipchain
- 1.3kLOC
- Main functions with APIs
  - Genesis Policy Request
  - New Policy Request
  - Get Policy Request
  - Verify Policy Request

Evaluation

Testbed: 32-core Intel Xeon CPU at 2.6 GHz with 66GB of RAM (one server of IC cluster)



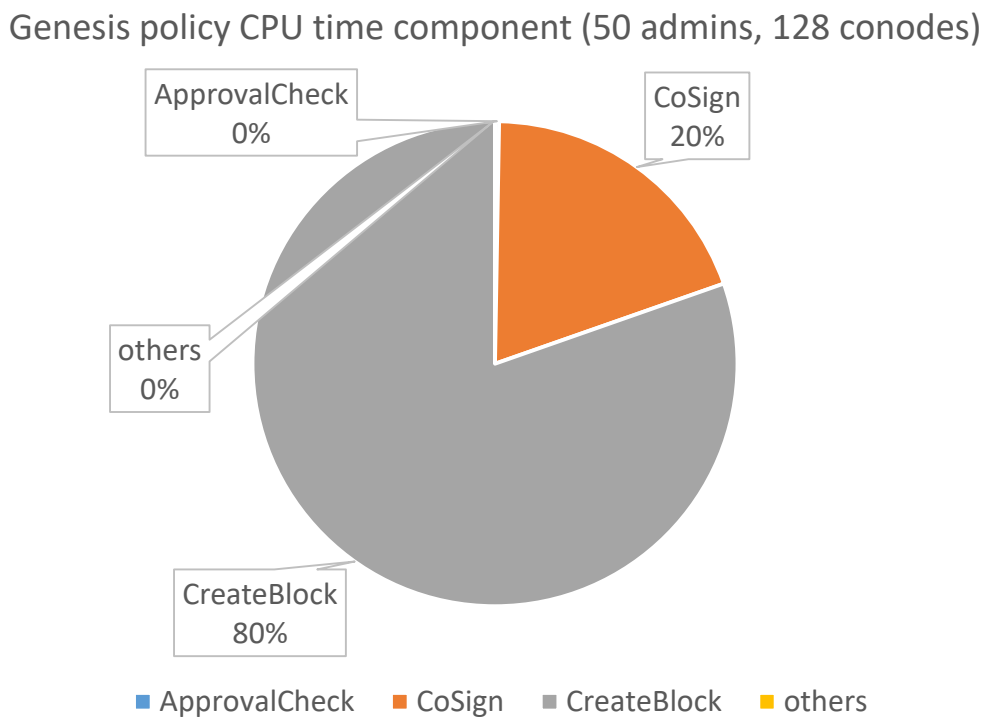


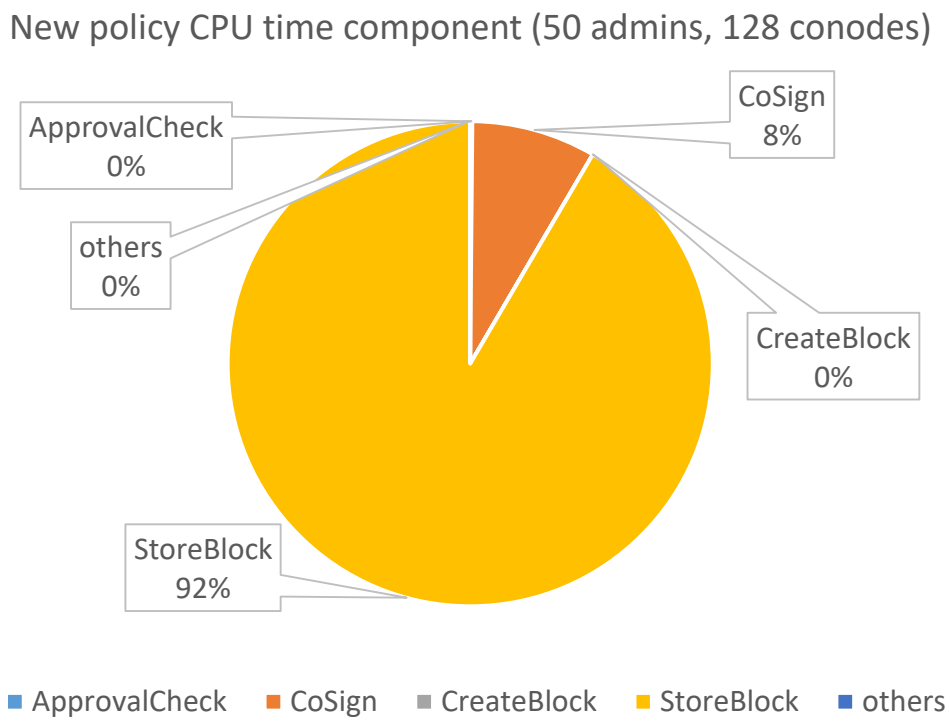
Maximum 0.18 sec for 100 admins

Maximum 20.8 sec for 128 conodes

Evaluation

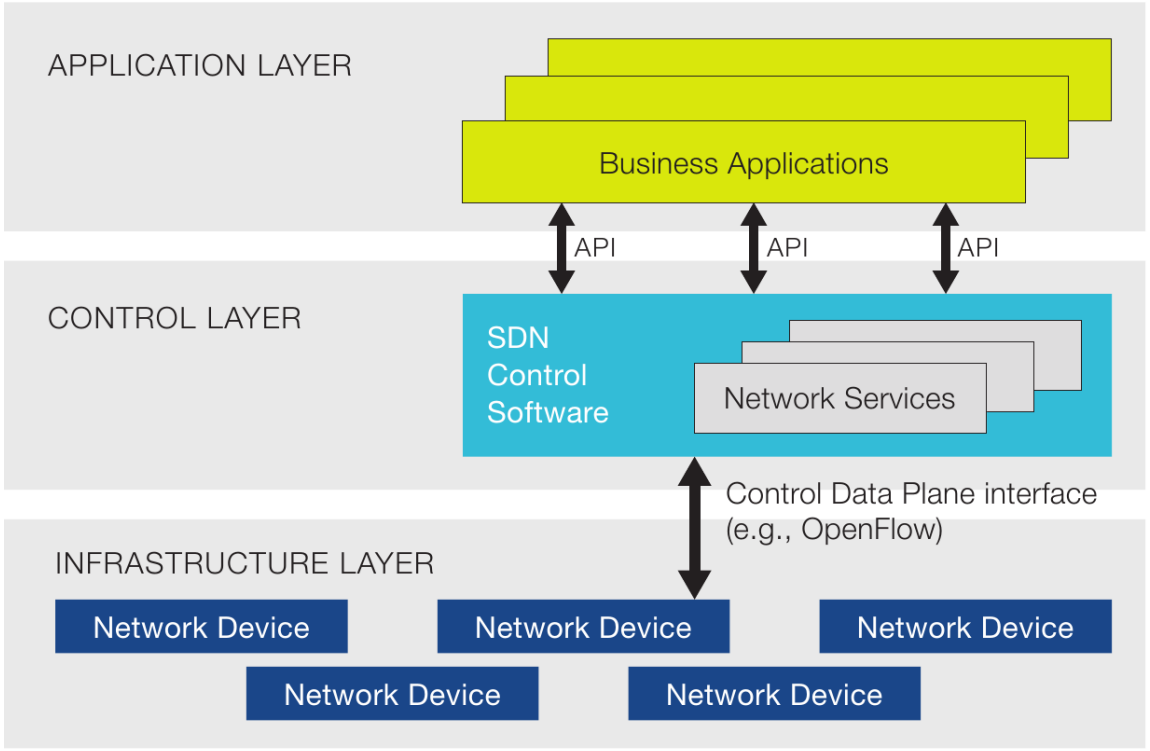
Time cost component





Compared to SDN

- Follower routers can be seen as SDN controllers
- Security-enhanced SDN application layer
- Easy to rollback to a previous correct network configuration



[3] https://www.sdxcentral.com/sdn/definitions/inside-sdn-architecture/

Future Work

- Performance evaluation & analysis
  - Tendency and limit
  - Bandwidth
  - Time cost vs number of policies
- Protocol improvement
  - Multiple groups of admins
  - Hierarchical network policy

Thank you!

Implementation