2019/04/19 @ 상명대

Stellar Consensus Protocol 분석

발표자: 임연주(yeonjoo@pel.smuc.ac.kr)

최서윤(seoyun@pel.smuc.ac.kr)

지도교수: 이종혁(jonghyouk@smu.ac.kr)

상명대학교 프로토콜공학연구실

목 차

- 개요
- FBA(Federated Byzantine Agreement)
  - 쿼럼 슬라이스와 쿼럼
  - 시스템 안전성
- SCP(Stella Consensus Protocol)
  - Federated voting
  - Consensus Protocol
    - Nomination
    - Balloting
- 추후 계획

목 차

개요

- 개요
  - 2015년 David Mazières가 FBA(Federated Byzantine Agreement) 프로토콜 기반 합의 알고리즘인 SCP 공개
  - 현재 Stellar lumens(암호화폐)의 합의 알고리즘으로 사용
    - stellard에서 stellar-core으로 합의 알고리즘 변경
      - stellard: 리플의 합의 알고리즘 기반
      - stellar-core: SCP
  - 분산된 인터넷 인프라 구축을 위한 표준화 재정 중
    - 분산된 환경을 위해 SCP에 대한 연구가 진행되고 있음
      - IRTF DINRG(Internet Research Task Force Distributed Internet iNfrastructure Research Group)

개요

- 특징

1. Decentralized control
   - 모든 노드가 합의 과정 참여하며, 합의에 대한 제어를 중앙기관이
2. Flexible trust
   - 사용자의 신뢰 노드(쿼럼 슬라이스)를 선택할 자유를 제공
3. Low latency
   - 빠른 시간(3~5초) 내에 합의 도달
4. Asymptotic Security
   - 디지털 서명, 해시함수의 매개변수를 조정하여 안전성 제공

목 차

- 개요
- FBA(Federated Byzantine agreement)
  - 쿼럼 슬라이스와 쿼럼
  - 시스템 안전성
- SCP(Stella Consensus Protocol)
  - Federated voting
  - Consensus Protocol
    - Nomination
    - Balloting
- 추후 계획

FBA(Federated Byzantine Agreement)

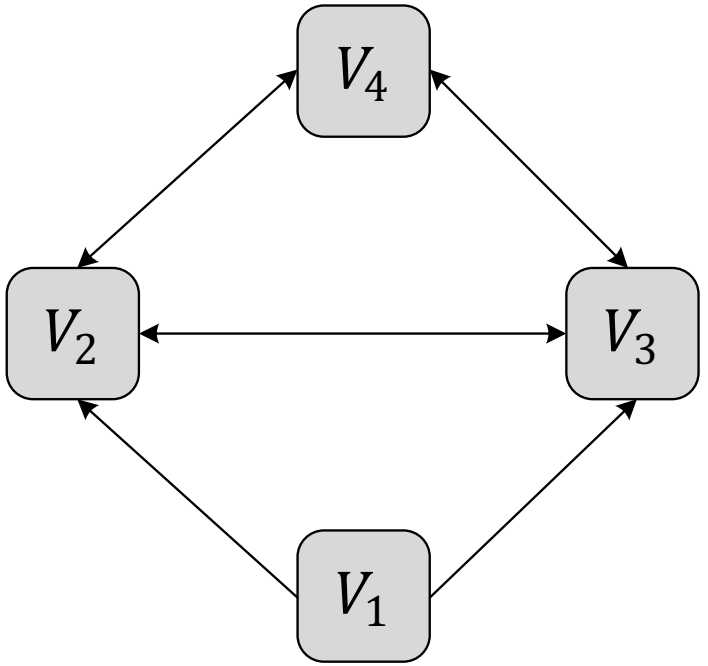
- FBA(Federated Byzantine Agreement)
  - 개념
    - 리소스 기반의 합의 참여 조건이 없이 모든 노드가 합의 과정에 참여
      - 기존 PoW의 경우 해시 연산, PoS의 경우 담보금 등의 리소스를 기반으로 합의에 참여할 수 있었음
    - 각 노드는 네트워크 참여 시, 신뢰할 노드를 선택
      - 신뢰 노드 선택에 따라 자신의 쿼럼 슬라이스가 형성됨
    - 합의의 최소 단위인 쿼럼을 자신이 형성

FBA(Federated Byzantine Agreement)

- 쿼럼 슬라이스와 쿼럼
  - 쿼럼 슬라이스 (Quorum slice)
    - 자신과 자신이 신뢰하는 노드들의 집합
      - 최소 노드 수: 2
    - 각 노드가 신뢰하는 노드를 직접 선정
  - 쿼럼(Quorum)
    - 자신의 쿼럼 슬라이스와 쿼럼 슬라이스 내 노드들의 쿼럼 슬라이스들의 합집합
      - 최소 쿼럼 슬라이스 수: 1
      - 최소 노드 수: 3
    - 합의에 도달하는데 충분한 노드 집합

FBA(Federated Byzantine Agreement)

- 쿼럼 슬라이스와 쿼럼
  - 쿼럼 슬라이스의 다이어그램
    - v_i = 노드
    - $Q(v_i)$ = 노드 v_i 의 쿼럼 슬라이스
    - $\longrightarrow$ = 쿼럼 슬라이스 종속성(신뢰 노드)

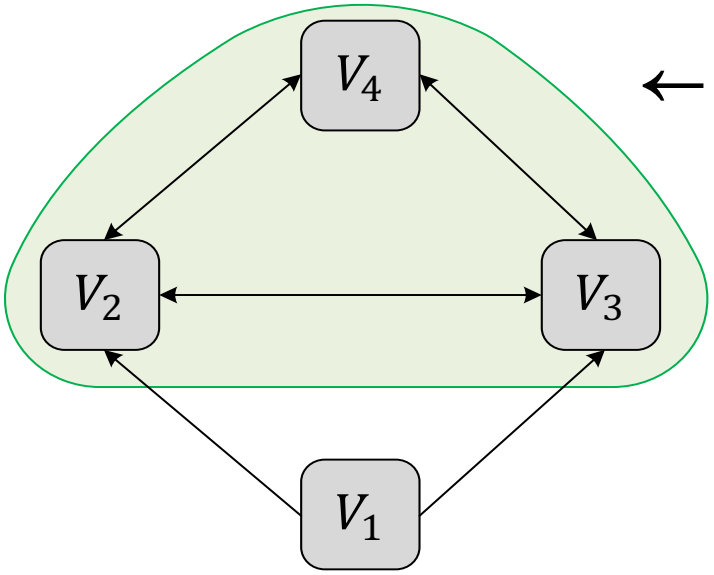


$$Q(v_1) = \{\{v_1, v_2, v_3\}\}$$

$$Q(v_2) = Q(v_3) = Q(v_4) = \{\{v_2, v_3, v_4\}\}$$

FBA(Federated Byzantine Agreement)

- 쿼럼 슬라이스와 쿼럼



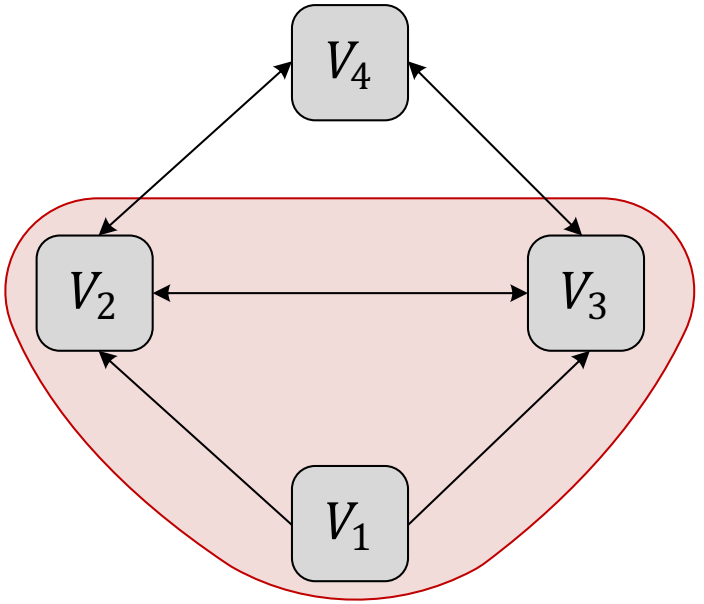
← 노드 v_2, v_3, v_4 의 쿼럼, 각 노드들의 쿼럼 슬라이스

$Q(v_1) = \{\{v_1, v_2, v_3\}\}$

$Q(v_2) = Q(v_3) = Q(v_4) =$

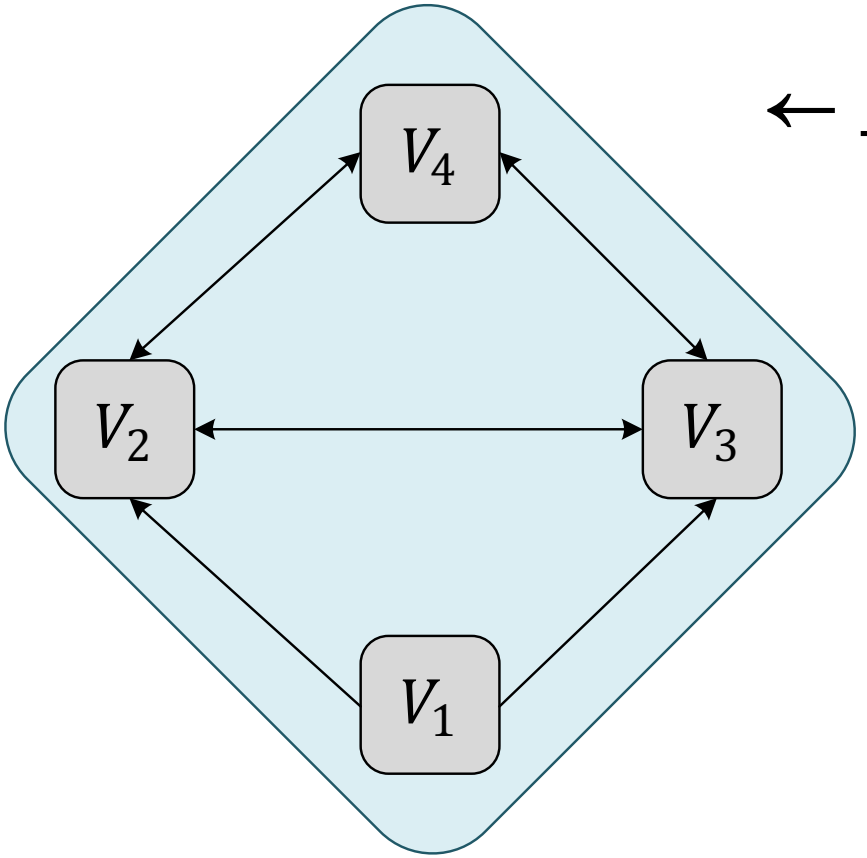
$\{\{v_2, v_3, v_4\}\}$

노드 v_1 의 쿼럼 슬라이스 ,쿼럼은 아님 →



FBA(Federated Byzantine Agreement)

- 쿼럼 슬라이스와 쿼럼



← 노드 v_1을 포함하는 최소의 쿼럼

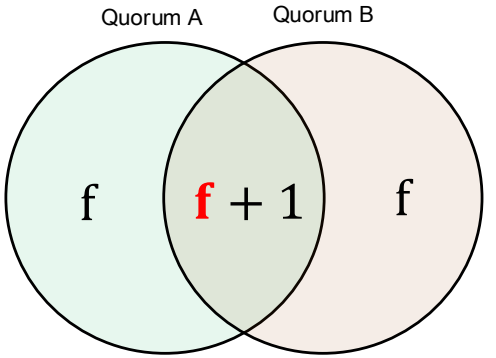
$$Q(v_1) = \{\{v_1, v_2, v_3\}\}$$

$$Q(v_2) = Q(v_3) = Q(v_4) = \{\{v_2, v_3, v_4\}\}$$

FBA(Federated Byzantine Agreement)

- FBA(Federated Byzantine Agreement)
  - 모든 노드간의 합의
    - 쿼럼의 크기 $= 2f + 1$
    - 두 개의 쿼럼이 교차하는 경우, 적어도 $f + 1$ 이상의 교집합을 가져야 함
      - 예시
        - f = 비잔틴 행위를 하는 노드 (ill-behaved node)
        - $\mathrm{N} - f$ = 올바른 노드 (well-behaved node)

$\mathrm{N} = 3\mathrm{f} + 1$



- **f** : ill-behaved node

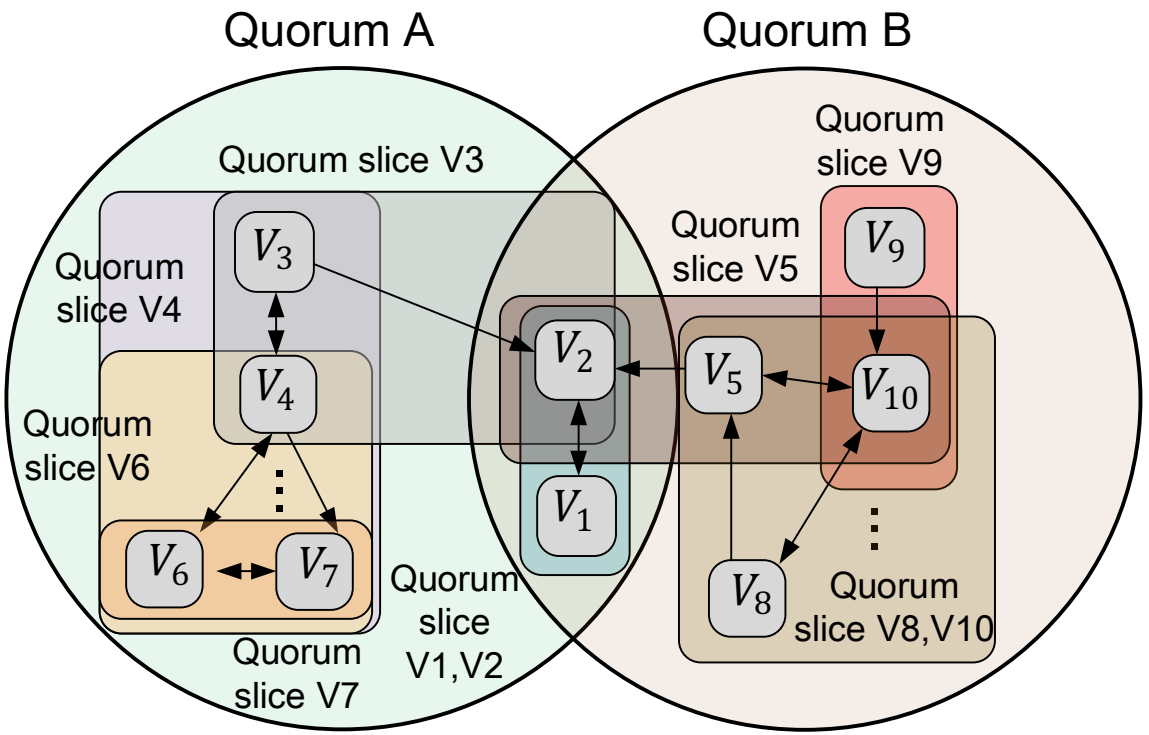
    - 두 쿼럼 사이의 모순된 결론을 내리지 않을 수 있음

FBA(Federated Byzantine Agreement)

- 시스템 안전성
  - 쿼럼 교차(Quorum Intersection)
    - 개념
      - 쿼럼들 중 어떤 두 쿼럼이 노드 하나를 공유하는 필요충분조건이 충족되는 경우를 쿼럼 교차라고 함
    - 쿼럼 간의 교차되는 부분을 통해 안정성 제공
      - 다른 쿼럼 간의 의견 합의를 이룰 수 있도록 함
      - 교차되지 않은 쿼럼에 대한 합의의 안전은 보장하지 않음

FBA(Federated Byzantine Agreement)

- 시스템 안전성
  - 쿼럼 교차(Quorum Intersection)
    - 예제 그림



주체	신뢰 노드
V_1	V_2
V_2	V_1
V_3	V_4, V_2
V_4	V_3, V_6, V_7
V_5	V_2, V_{10}
V_6	V_7, V_4
V_7	V_6
V_8	V_5, V_{10}
V_9	V_{10}
V_{10}	V_5, V_8

FBA(Federated Byzantine Agreement)

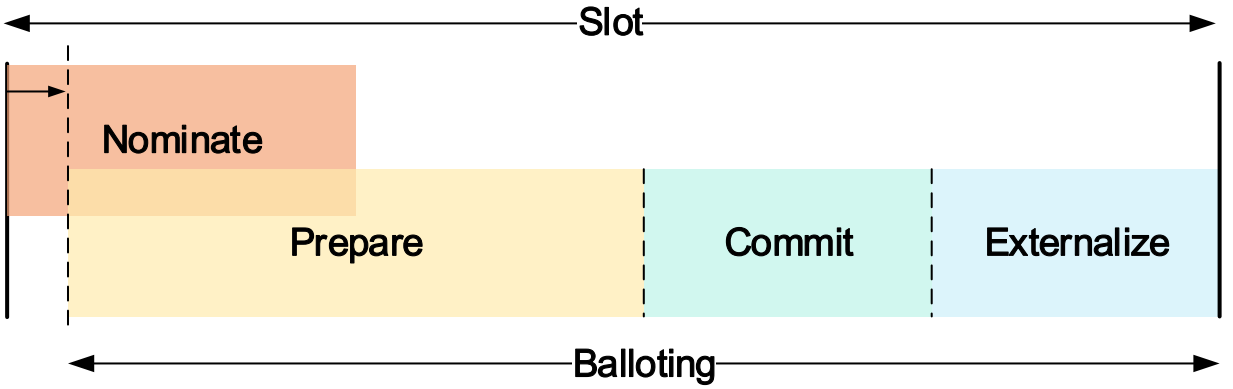
- 시스템 안전성
  - Dset(Dispensible set)
    - 특정 노드 집합에 완전히 영향을 미칠 수 있는 노드 집합
    - 시스템 구축 시, Dset이 정해짐
      - 실행 동안 노드의 행동 모니터링
  - Dset 예제 그림 추가

목 차

- 개요
- FBA(Federated Byzantine Agreement)
  - 쿼럼 슬라이스와 쿼럼
  - 시스템 안전성
- SCP(Stella Consensus Protocol)
  - Federated voting
  - Consensus Protocol
    - Nomination
    - Balloting
- 추후 계획

SCP(Stellar Consensus Protocol)

- 메인 서브루틴
  - Federated voting(연합형 투표)
    - 투표 진행에 있어 점진적 상태 변환을 위한 투표
      - vote -> accepted -> confirmed
    - 이분법적인(bivalent) 투표 진행
      - “a”, “!a”
    - Externalize를 제외한 모든 단계(nomination, prepare, commit)마다 Federated voting 수행

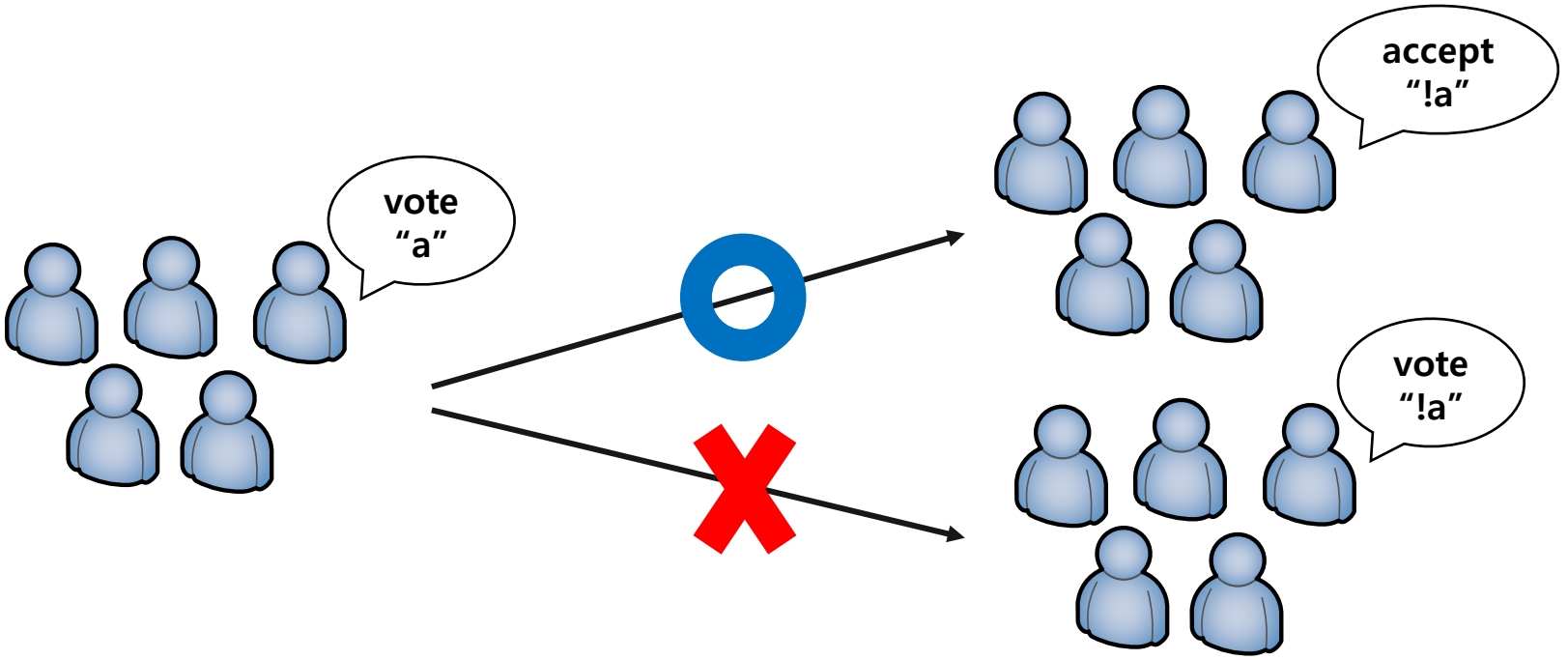


SCP(Stellar Consensus Protocol)

- Federated Voting(연합형 투표)
  - 메시지 종류
    - Vote
    - Accept
    - Vote or accept
    - Confirm

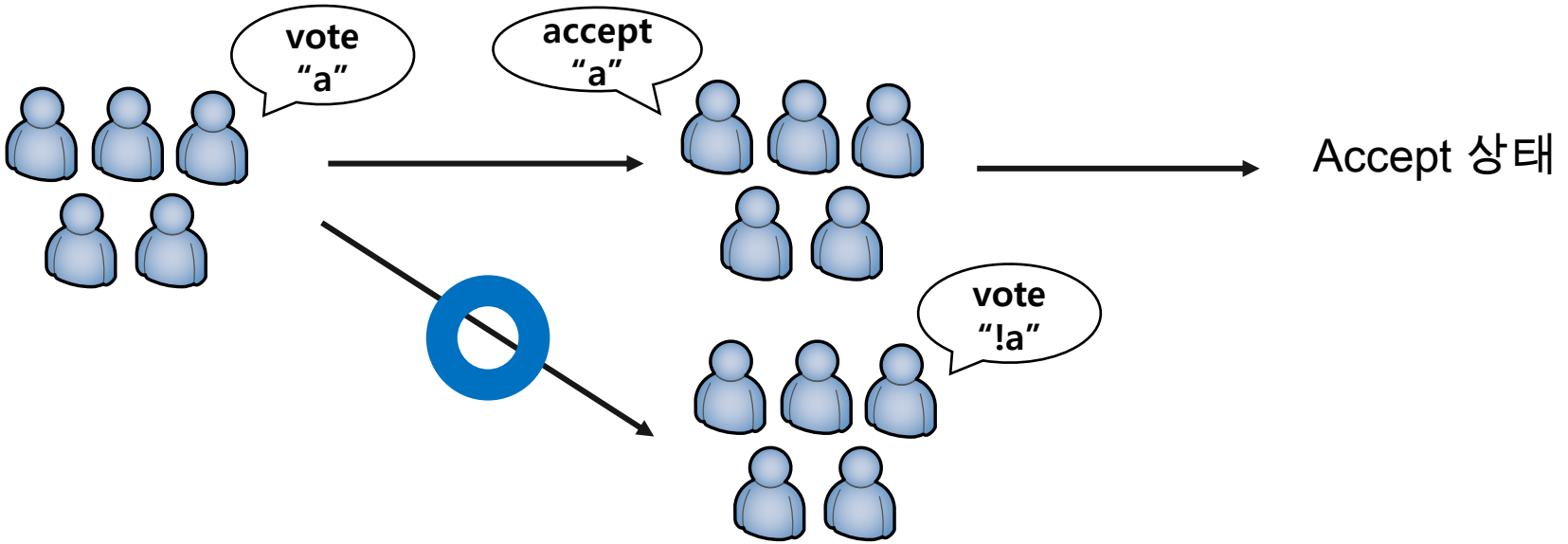
SCP(Stellar Consensus Protocol)

- Federated Voting(연합형 투표)
  - 메시지 종류
    - Vote
      - Vote (a)는 “a”를 후보에 등록할 것에 대한 주장을 의미
      - 이후 !a에 대해 accept은 할 수 있지만 vote a!는 절대 불가



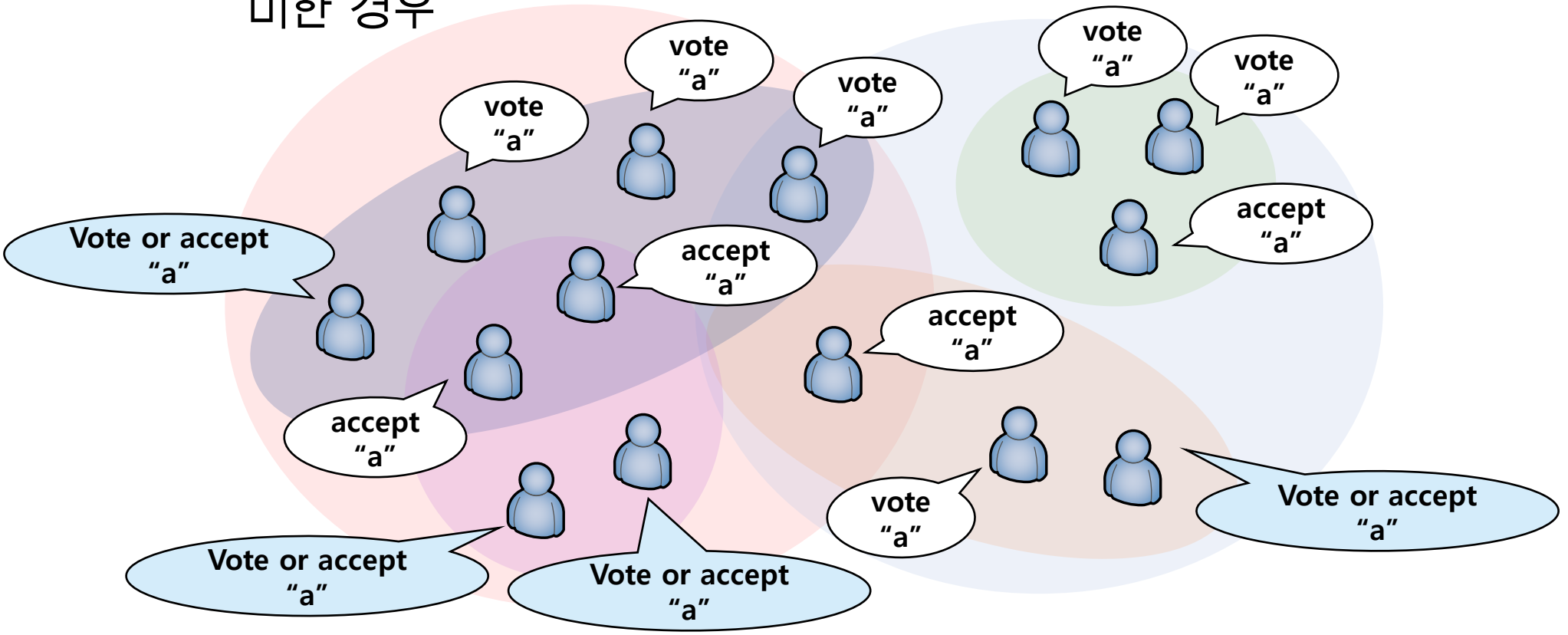
SCP(Stellar Consensus Protocol)

- Federated Voting(연합형 투표)
  - 메시지 종류
    - Accept
      - Vote된 “a”에 대해 “a”를 후보로 선정한다는 것에 동의함을 의미
      - Vote하지 않았다면 무조건 accept을 전송해야 함
        - 비잔틴 합의 구조를 유지하기 위해 모든 노드가 투표에 참여할 것이라고 가정
      - Vote !a 를 전송했더라도 “a”에 accept 할 수 있음



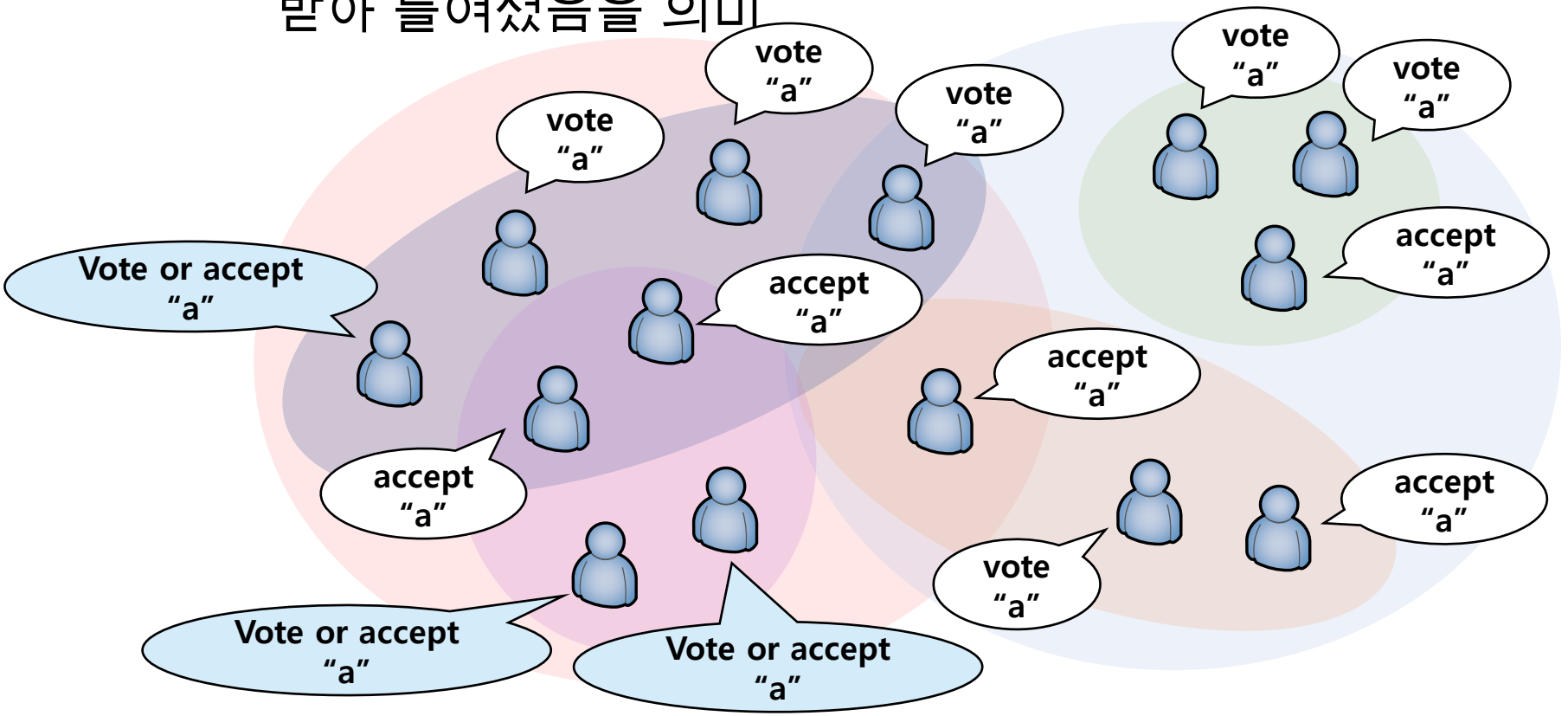
SCP(Stellar Consensus Protocol)

- Federated Voting(연합형 투표)
  - 메시지 종류
    - Vote or accept
      - 임의의 노드가 투표하지 않은 상태에서 이미 vote나 accept이 무의미한 경우



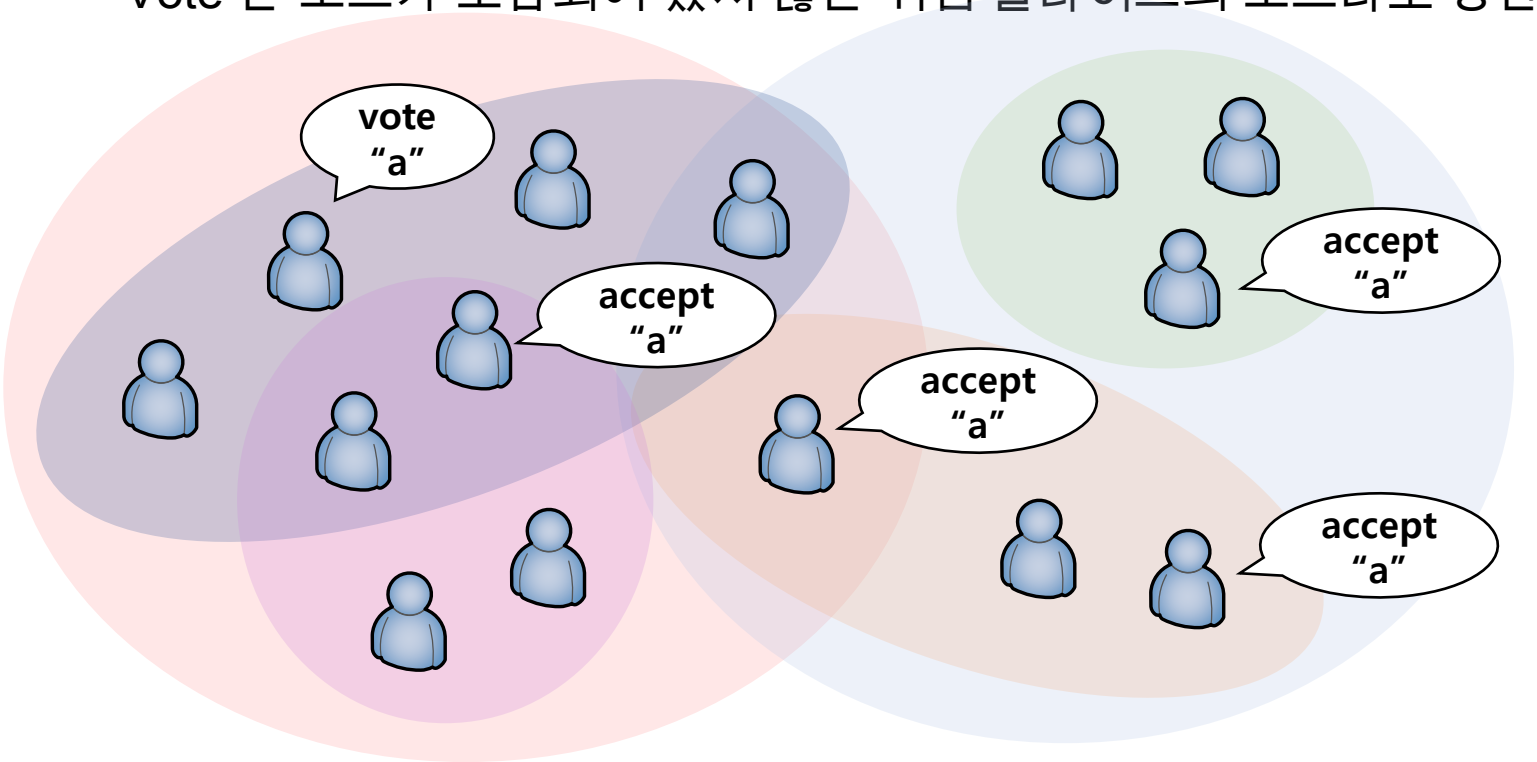
SCP(Stellar Consensus Protocol)

- Federated Voting(연합형 투표)
  - 메시지 종류
    - Confirm
      - 모든 노드가 “a”에 대해 승인하는 메시지를 전송했으며 “a”가 후보로 받아 들여졌음을 의미



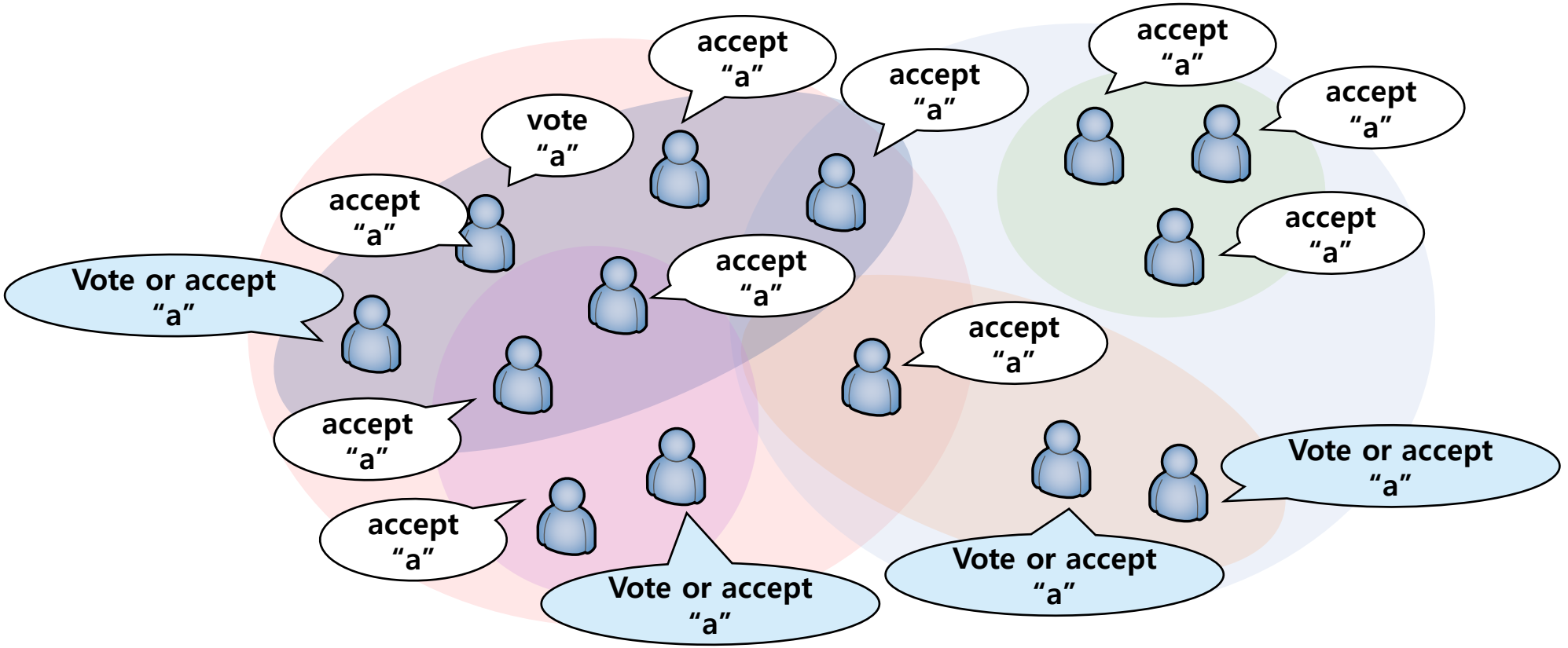
SCP(Stellar Consensus Protocol)

- Federated Voting(연합형 투표)
  - 임계 값
    - Blocking threshold (차단 임계 값)
      - 쿼럼의 슬라이스마다 하나의 노드라도 “accept” 한 경우
        - Vote 한 노드가 포함되어 있지 않은 쿼럼 슬라이스의 노드라도 상관 없음



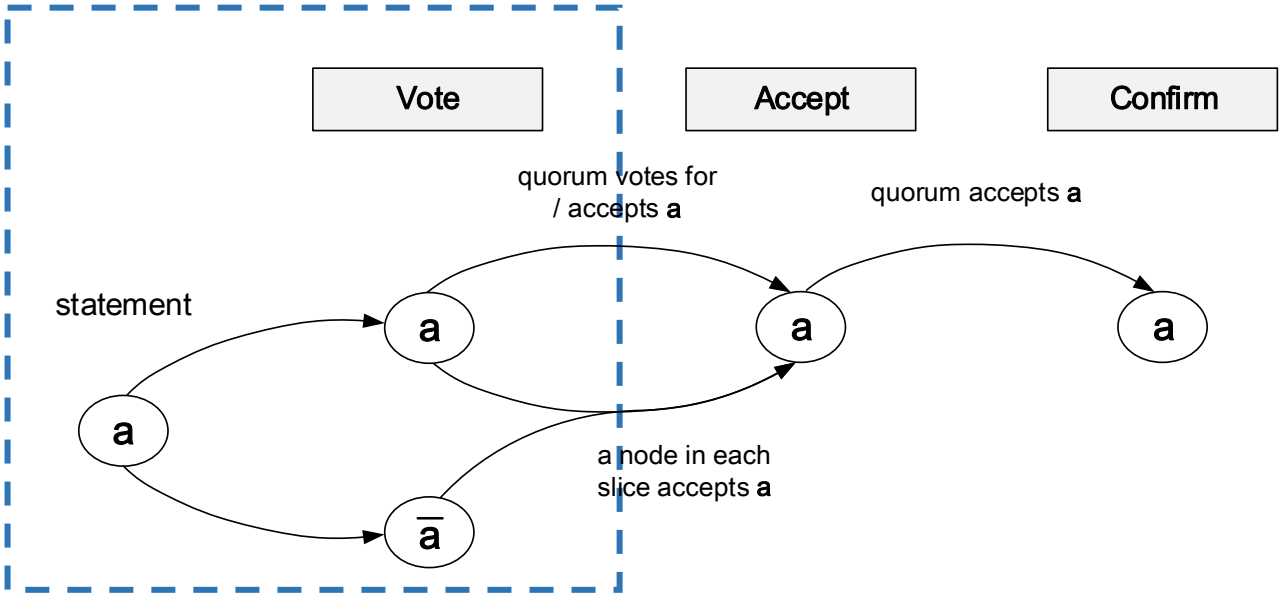
SCP(Stellar Consensus Protocol)

- Federated Voting(연합형 투표)
  - 임계 값
    - Quorum threshold (쿼럼 임계 값)
      - 쿼럼의 모든 노드가 “m”를 전송한 경우



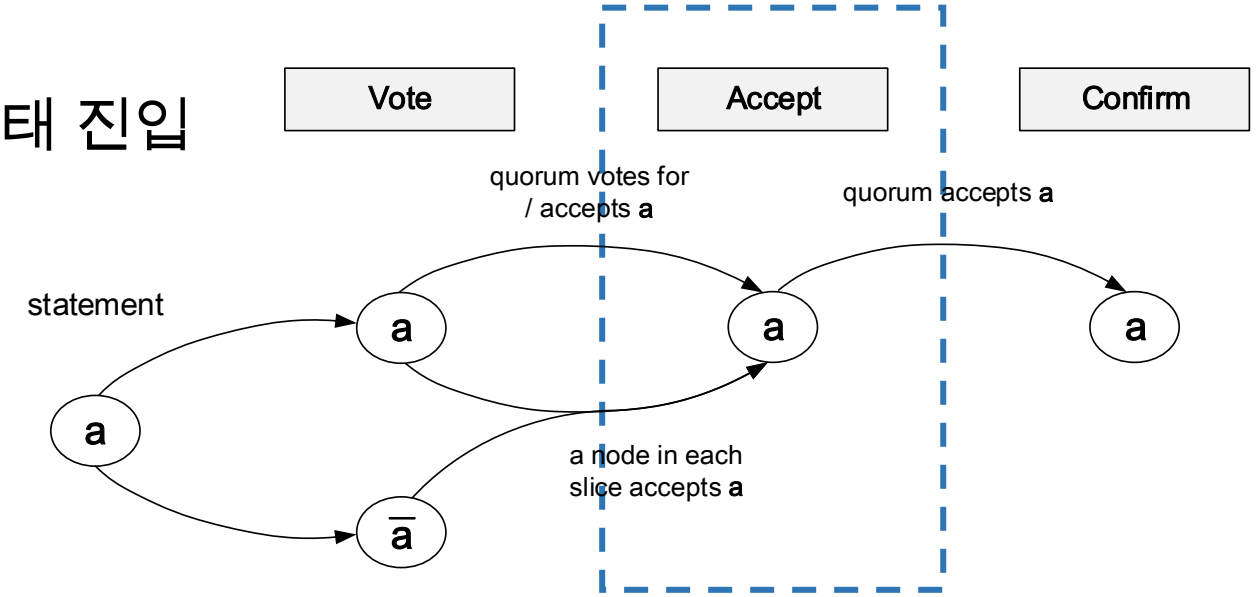
SCP(Stellar Consensus Protocol)

- Federated Voting(연합형 투표)
  - 상태
    - uncommitted or vote
      - 노드는 vote를 할 수도 안 할 수도 있음
        - 어떠한 값이 vote되었다면 voted
        - 어떠한 값도 vote 되지 않았다면 uncommitted



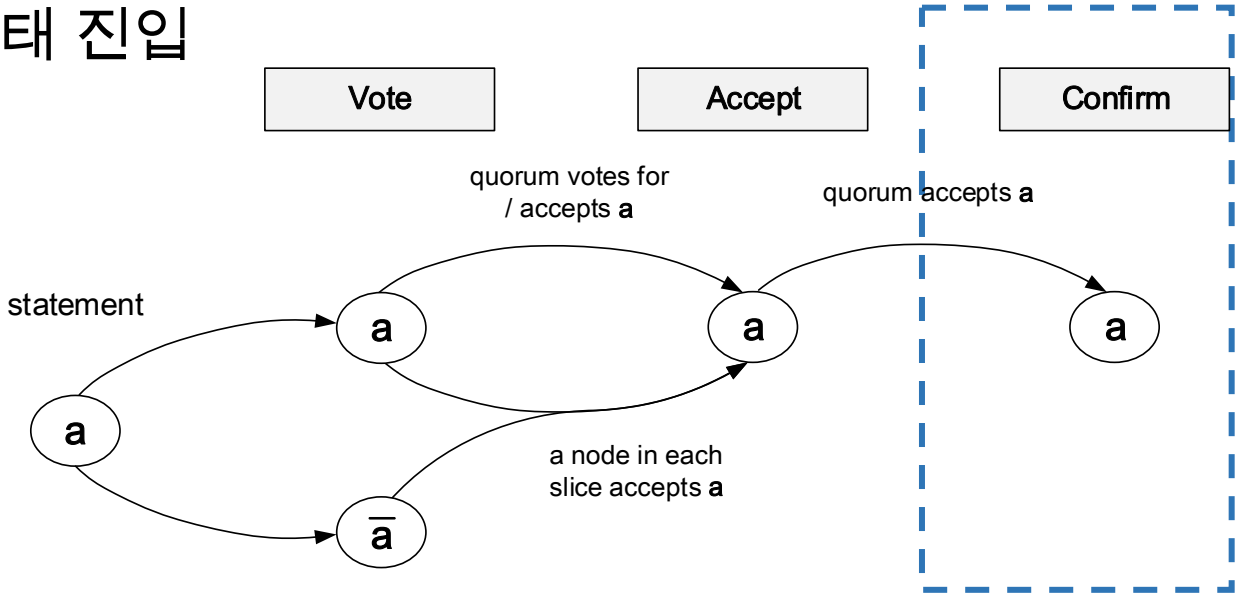
SCP(Stellar Consensus Protocol)

- Federated Voting(연합형 투표)
  - 상태
    - accepted
      - 쿼럼의 쿼럼 슬라이스 중 하나의 노드라도 “accept” 했고
        - (blocking threshold)
      - 쿼럼의 모든 노드들이 “accept” 또는 “vote-or-accept” 했다면
        - (quorum threshold)
      - ➔ a가 accepted 상태 진입



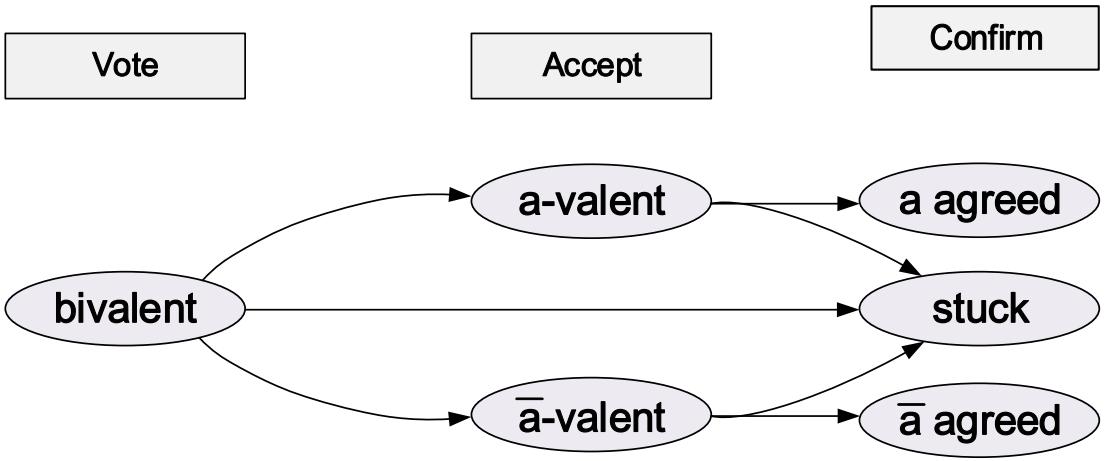
SCP(Stellar Consensus Protocol)

- Federated Voting(연합형 투표)
  - 상태
    - confirmed
      - 쿼럼의 모든 노드들이 “accept” 했다면
        - (quorum threshold)
        - “a”가 후보로서 accepted 됐다는 사실에 다시 한 번 더 승인
      - ➔ a가 confirmed 상태 진입



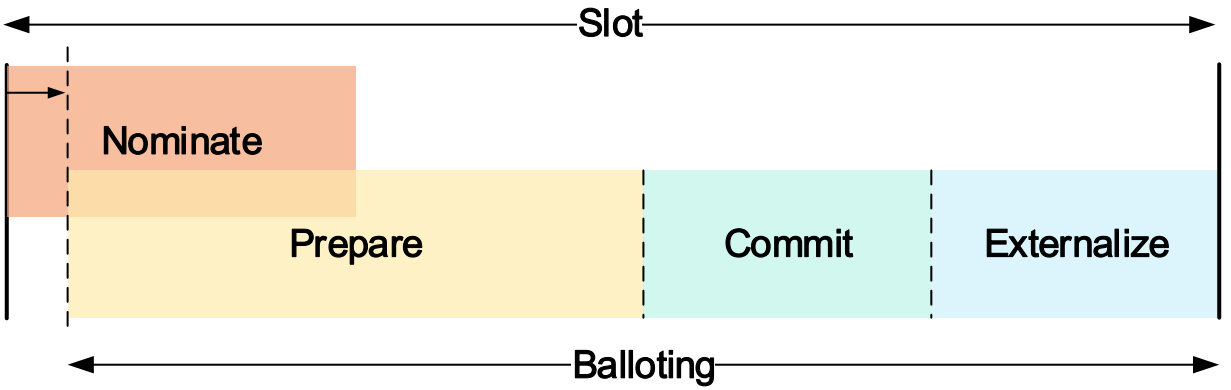
SCP(Stellar Consensus Protocol)

- Federated Voting(연합형 투표)
  - 결과 도출
    - Valent
      - 노드가 a/!a를 accept하면 각각 a-valent, !a-valent
    - Stuck
      - 하나 이상의 정상 노드들이 값을 도출해낼 수 없는 상태
        - e.g., a에 대한 vote와 a!에 대한 vote 수가 동일한 경우, timeout 이상의 시간이 소요된 경우
        - 카운터를 1증가시켜 재투표 진행
    - Agree
      - Accept 메시지에 동의함



SCP(Stellar Consensus Protocol)

- Consensus protocol
  - Nomination 매커니즘
    - 후보자 등록
  - Balloting 매커니즘
    - 당선자 투표
    - 3 단계로 진행됨
      1. Prepare
      2. Commit
      3. Externalize



SCP(Stellar Consensus Protocol)

- Nomination Mechanism
  - Federated voting을 통해 후보자(candidate values) 등록
    - 일정 시간동안 발생한 여러 개의 후보 값이 하나의 값으로 결합되기까지의 단위를 라운드(round) 라고 함
      - 노드는 항상 round 1에서 nomination 시작
      - 타임아웃 시간: n+1초
        - n+1초 동안 nominate 된 값이 확인되지 않은 경우 n+1 라운드로 진행
    - Priority가 높은 노드들과 똑같이 nominate하는 것을 지향
      - 이웃들(neighbors) 중 우선순위(priority)가 가장 높은 노드로부터 이전 라운드와 현재 라운드의 votes를 echo함
        - Repeat, follow
      - 우선순위가 높은 노드들을 기준으로 동기화 됨

SCP(Stellar Consensus Protocol)

- Nomination Mechanism

*$|S|$ = cardinality
(the number of elements in set S)

*i = slot number

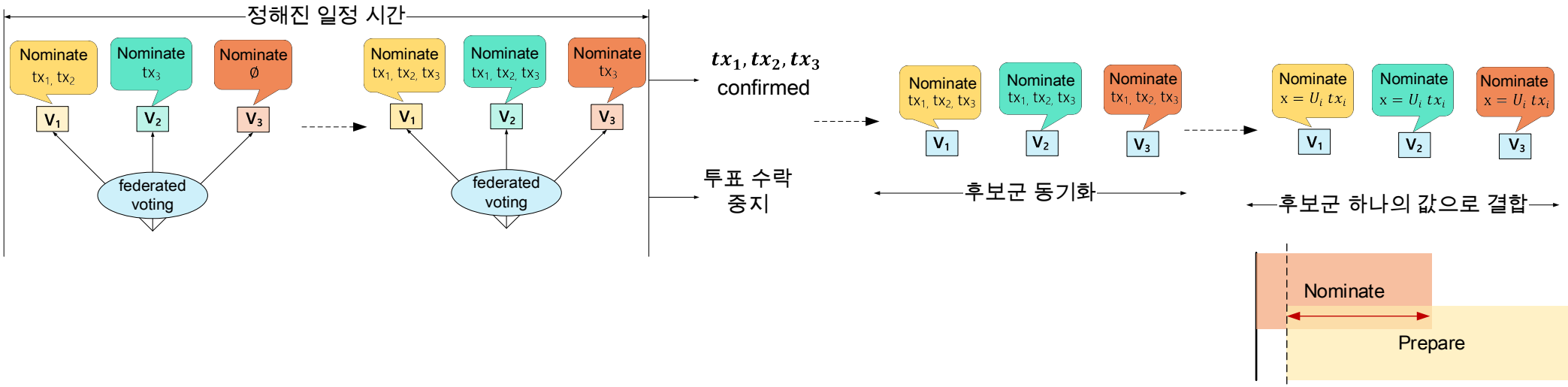
*n = round number

*v = each node

  - Weight(v, v') = $\frac{|\{q|q \in Q(v) \wedge v' \in q\}|}{|Q(v)|}$
  - Neighbors(v, n) = $\{v'|G_i(N, n, v') < h_{max} \cdot weight(v, v')\}$
    - h_{max}= 2^{256}
    - G는 공개키를 의미
      - 스텔라 루멘의 공개키 범위 내에서 첫 바이트 시작은 무조건 G
  - Priority(n, v') = $G_i(P, n, v')$
  - N, P are both 32bit XDR(External Data Representation) values
    - N = SHA-256(1 || n || v)
    - P = SHA-256(2 || n || v)

SCP(Stellar Consensus Protocol)

- Nomination Mechanism
  - 일정 시간동안 모인 후보 값들을 하나의 값으로 결합
    - 결합된 하나의 값을 “후보 값(candidate value)”
      - White paper
        - Gi(m) = SHA-256(i || x_(i-1) || m)
      - Draft
        - Gi(m) = SHA-256(i || m)



SCP(Stellar Consensus Protocol)

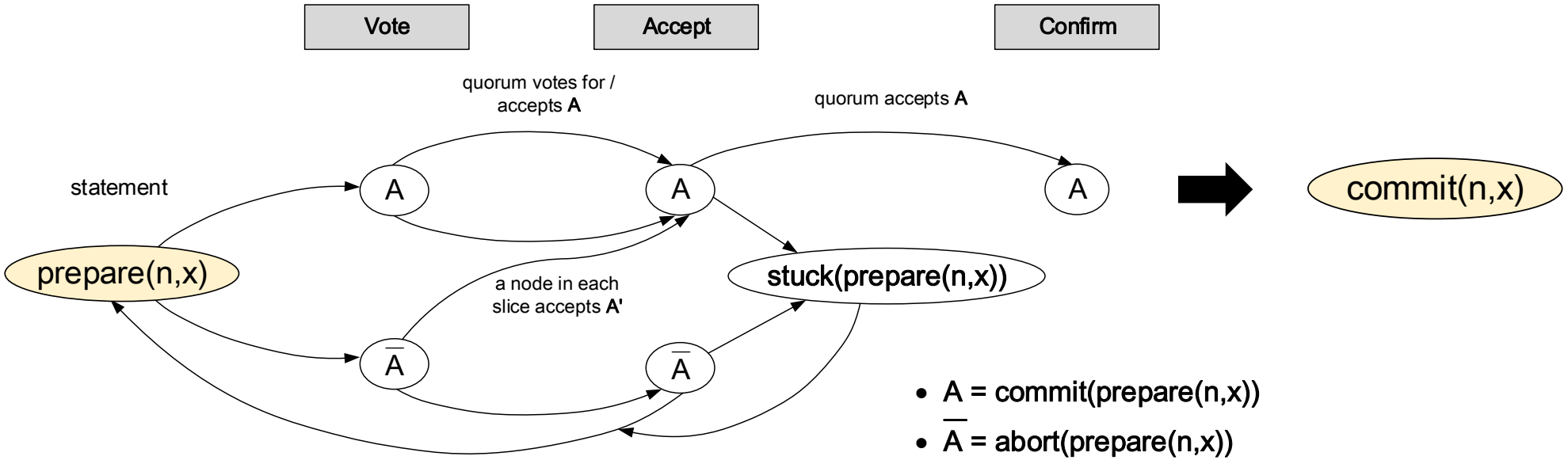
- Balloting Mechanism
  - 3 단계를 통해 투표를 진행
    1. PREPARE
       - 후보군(Candidate values)에 대한 투표 용지 준비(Prepare)
       - 하나의 후보가 선정 될 때까지 prepare 과정을 반복
         - 최종 선정: accept(commit(x))
    2. COMMIT
       - 준비된(prepared) 후보 값에 대한 투표
    3. Externalize
       - 최종 선정을 알림

SCP(Stellar Consensus Protocol)

- Balloting - PREPARE
  - Prepare에서의 Federated Voting
    - Federated Voting
      - Vote 메시지
        - commit(x): x에 대한 수용(찬성)을 의미
        - abort(x): x에 대한 중단(반대)을 의미
      - Accept 메시지
        - commit(prepare(n,x))
          - Quorum threshold 에 부합할 시, Accpet(commit(prepare(n,x))) 로 넘어감
        - abort(prepare(n,x))
          - Quorum threshold 에 부합할 시, prepare(n+1,x)로 재투표 진행
      - Confirm
        - x 의 투표 용지가 준비 되었음을 의미
          - prepared(n,x)
        - prepare 메시지가 confirm 됨과 동시에 commit(n,x) 상태로 넘어감

SCP(Stellar Consensus Protocol)

- Balloting - PREPARE
  - Prepare에서의 Federated Voting

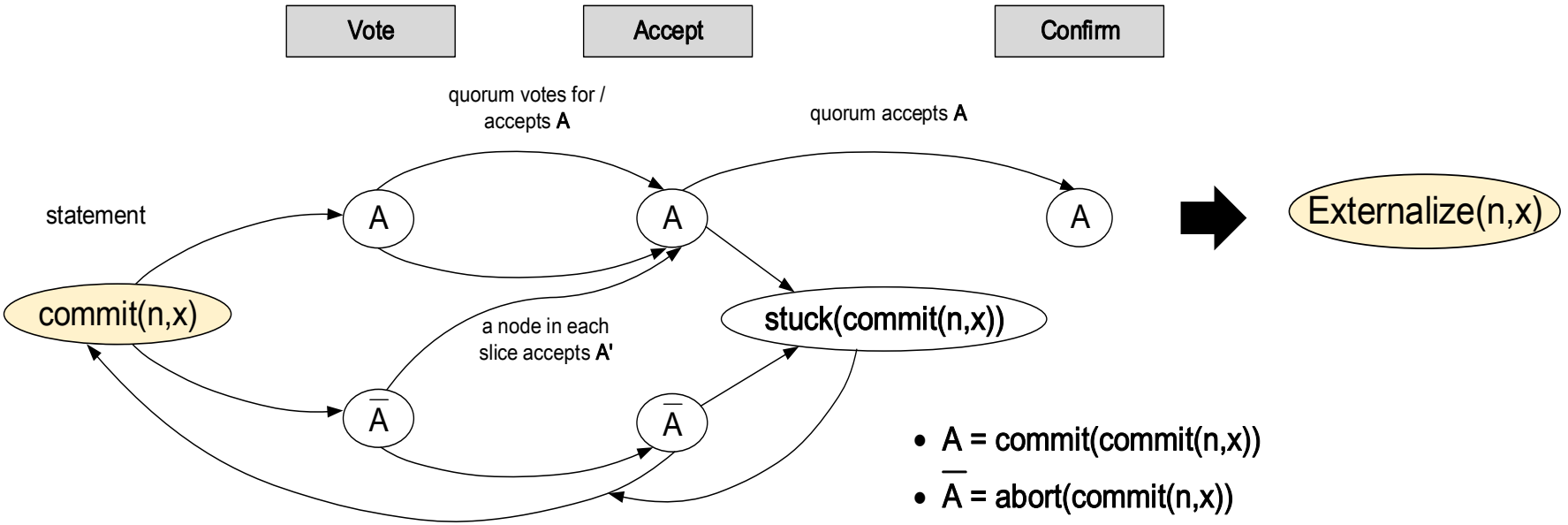


SCP(Stellar Consensus Protocol)

- Balloting - COMMIT
  - Commit에서의 Federated Voting
    - Vote
      - 준비된 후보에 대한 찬반 투표를 진행
        - commit(x): x에 대한 수용(찬성)을 의미
        - abort(x): x에 대한 중단(반대)을 의미
    - Accept
      - commit(commit(n,x))
        - Quorum threshold 에 부합할 시, Accpet(commit(commit(n,x))) 로 넘어감
      - abort(commit(n,x))
        - Quorum threshold 에 부합할 시, commit(n+1,x)로 재투표 진행
    - Confirm
      - x 가 ballot n에서 최종 당선되었음을 의미
      - commit 메시지가 confirm 됨과 동시에 Externalize 단계로 넘어감

SCP(Stellar Consensus Protocol)

- Balloting – EXTERNALIZE
  - Externalize
    - 투표 결과 공표



추후 계획

- IoT firmware update
  - IETF suit(software updates for Internet of Things)
    - draft-ietf-suit-architecture 제안을 위한 연구
    - 관련 논문 분석
      - Lee, Boohyung, and Jong-Hyouk Lee. "Blockchain-based secure firmware update for embedded devices in an Internet of Things environment." *The Journal of Supercomputing* 73.3 (2017): 1152-1167.

감사합니다!

백업1 - SCP 메시지 포맷

- Vote Message
  - Statement 구조체
    - `typedef opaque Hash[32];  // SHA-256`
    - `struct SCPStatement {`

```
        PublicKey      nodeID;        // v (node signing message)
        unit64         slotIndex;
        Hash           quorumSetHash;
        SCPStatement   pledges;
    }
```

    - `typedef opaque Signature<64>;`
    - `struct SCPEnvelope  {`

```
        SCPStatement   statement;
        Signature      signature;
    }
```

백업1 - SCP 메시지 포맷

- Nomination
  - Nominate 메시지 포맷
    - Typedef opaque Value<>;
    - struct SCPNominate {
      Value voted<>; // vote to nominate these values
      Value accepted<>; // assert that these are accepted
      }
    - union SCPStatement switch (SCPStatementType type) {
      case SCP_ST_NOMINATE:
      SCPNomination nominate;
      };

백업1 - SCP 메시지 포맷

- Balloting
  - Ballot 구조체
    - ```
      struct SCPBallot {
            unit32 counter;       // n: 투표 용지수
            Value  value;         // x: 후보 값
          }
      ```
      ```

백업1 - SCP 메시지 포맷

- Balloting
  - Prepare 메시지 포맷
    - struct SCPPrepare {

```
        SCPBallot      ballot;        // b.n, b.x
        SCPBallot      *prepared;     // p.n, p.x
        uint32         aCounter;
        uint32         hCounter;
        uint32         cCounter;
    }
```

백업1 - SCP 메시지 포맷

- Balloting
  - Commit 메시지 포맷
    - ```
      struct SCPCommit    {
            SCPBallot     ballot;
            uint32        preparedCounter;
            uint32        hCounter;
            uint32        cCounter;
        }
      ```
      ```

백업1 - SCP 메시지 포맷

- Balloting
  - Externalize 메시지 포맷
    - ```
      struct SCPExternalize      {
            SCPBallot      commit;
            uint32         hCounter;
        }
      ```
      ```