PBFT Consensus Algorithm

- 블록체인에서 합의 알고리즘은 Byzantine General Problem과 Fork 문제를 해결하기 위한 방안
  - Byzantine General Problem
    - 블록체인 뿐만이 아니라 분산 환경에서 항상 나타나는 문제
    - 네트워크 내에 배신자가 있더라도 합의 내용에 문제가 없어야 한다.

- FLP Impossibility
  - 기존 분산환경에서 어떤 합의 알고리즘이 네트워크에서 통용되기 위해서는 Safety와 Liveness라는 특성을 가지고 있어야 한다.

    - Safety : 노드 간 합의가 발생했다면, 어느 노드가 접근하든 그 값은 동일해야 한다. (블록체인의 finality)
    - Liveness : 합의 대상(transaction or block)에 문제가 없다면, 네트워크내에서 반드시 합의가 이루어진다.

    - 기존 분산환경에서의 Safety와 Liveness의 관계
      - Safety를 충족하는 상황에서 어떻게 하면 Liveness의 손실을 최소화할 수 있는가?
      - Safety > Liveness

    - 블록체인에서의 Safety와 Liveness의 관계
      - 모든 거래를 가감없이 수용하는 구조를 채택하여 Liveness를 극대화한다.
      - 하드포크가 발생할 경우 "최장 길이 체인 선택 알고리즘"을 통해 Safety문제 보완
      - '확률적으로'Safety, 블록체인에서는 Finality를 확보하였다.

  - 비동기 네트워크 내에서는 Safety와 Liveness를 모두 완벽하게 만족하는 합의 알고리즘을 설계하는 것이 불가능하다는 것이 증명되었다. 이 증명을 "FLP Impossibility"라고 한다.

    => 비동기 네트워크에서는 어떤 한 노드에서 문제가 발생했을 경우 그 노드에서 합의가 됐는데 단순히 응답에 오랜 시간이 걸리는 건지, 아니면 합의 과정에서 충돌이 발생해서 응답하지 않는건지 알 수 없기 때문

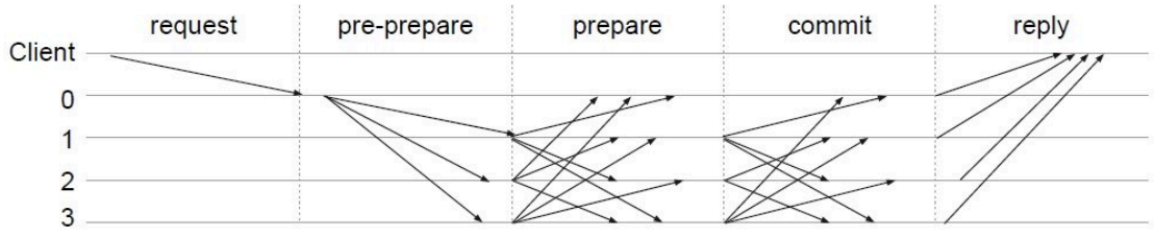
    => 블록체인에서 어떤 합의 알고리즘을 채택한다는 것 = Safety와 Liveness 중 어느것에 비중을 둘 것인가

- Concept
  - Practical Byzantine Fault Tolerance
  - Safety를 확보하고 Liveness를 일부 희생하면서, 비동기 네트워크에서도 합의릊 이룰 수 있는 알고리즘
    = 네트워크에 배신자 노드가 어느 정도 있다고 해도 네트워크 내에서 이루어지는 합의

의 신뢰를 보장
    - 현재 블록체인 합의 알고리즘 중 BFT 방식을 채택했다고 하는 경우 대부분 PBFT의 변형 알고리즘이다.

- 증명
    - 가정
        - 비동기 네트워크 | 배신자 노드 f 개
    - 위의 가정을 기반으로 할 경우 총 노드 개수가 3f+1 개 이상이면 해당 네트워크에서 이루어지는 합의는 신뢰 가능하다.
    - 네트워크의 모든 노드는 트랜잭션을 반영할 것인지 Prepared Certificate & Commit Certificate라는 두 번의 절차를 거쳐 결정한다.

- 과정



1. 클라이언트가 상태 변환을 요청하는 Request 메세지 “m” 을 Primary Node에 전송한다.
    - Primary Node : 처음 상태변환 요청을 받은 노드
    - Backup Node : Primary Node를 제외한 네트워크의 나머지 노드

2. Primary Node가 Request 요청을 받으면 먼저 Pre-prepare라는 절차를 시행한다.
   2.1 해당 request에 대응하는 Sequential number “N”을 생성한다.
   2.2 네트워크의 나머지 모든 노드에게 Pre-prepare 메세지를 전송한다.
   {
   메세지 구성 : <Pre-prepare, V, N, D(m)> // V = 메세지가 전송되는 대상 (View)
   // N = Sequential Number
   // D(m) = 요청 메세지 m의 요약본
   }

3. 네트워크 내 임의의 Backup 노드 i가 Pre-prepare 메시지를 받고, D(m)과 V,N이 서로 대응되는 값인지 검증합니다.
   3.1 대응되지 않는 값일 경우 : Pre-prepare 메시지를 수용하지 않는다.
   3.2 대응할 경우 : Prepare 메시지를 생성해 네트워크의 나머지 모든 노드에게 전송
   {
   Prepare 메시지 구성 : < Prepare, V, N, D(m), i > // i = 메시지를 검증한 Backup Node의 번호
   }

4. 각각의 노드는 Pre-prepare 메시지와 Prepare 메시지를 수집한다. 수집한 Pre-prepare 메시지 개수가 2f+1개 이고 Prepare 메시지가 2f개 이상인 경우 “Prepared Certificate”라고 부르며, 해당 노드는 “Prepared the request”상태가 됩니다.
   => 이 때 Pre-Prepare 메시지가 2f+1인 이유는 Prepare 메시지에 V, N, D(m) 이 답겨 있기 때문에 Prepare 메시지 안에 Pre-prepared 메시지가 담겨 있다고 보기 때문이다. 따라서 최초 Primary Node로 부터 받은 Pre-prepared 메시지 1개 + Prepared 메시지 2f개 = 2f + 1

5. “Prepared certificate” 조건을 만족한 노드는 네트워크의 모든 노드에게 Commit 메시지를 전송한다
   {
   Commit 메시지의 구성 : <commit, V, N, i>
   }

6. 각각의 노드는 Commit 메시지를 수집하고, Commit 메시지가 2f+1개가 모이면 해당 노드는 “Commit Certificate”상태가 된다.

7. “prepared certificate” + “commit certificate” = “committed certificate”은 클라이언트가 요청한 request를 수용해 상태 변화 함수를 시행한다. = 네트워크의 합의 알고리즘이 작동하여 합의를 도출한다.