이 일 구 교수 지도

석사학위 청구논문

PBFT 확장성 향상을 위한 분산 환경에서의 결함 노드 판단 및 대응 방법

2023

성신여자대학교 대학원

미래융합기술공학과

이 은 영

PBFT 확장성 향상을 위한 분산 환경에서의 결함 노드 판단 및 대응 방법

이 일 구 교수 지도

이 논문을 석사학위논문으로 제출함

2022년 11월

성신여자대학교 대학원
미래융합기술공학과
이 은 영

인 준 서

이은영의 석사학위 논문으로 인준함

2022년 11월

심사위원장 김 경 진 (서명 또는 인)

심 사 위 원 임 연 섭 (서명 또는 인)

심 사 위 원 이 일 구 (서명 또는 인)

성신여자대학교 대학원

논문 개요

PBFT(Practical Byzantine Fault Tolerant)는 분산 네트워크 환경에서 비의도적·의도적 장애를 일정 수준 허용하여 전체 네트워크의 합의를 달성할 수 있는 합의 알고리즘으로 높은 성능과 절대적 최종성을 보장한다. 그러나 합의 과정에서 발생하는 메시지 전송량으로 인해 네트워크의 규모가 증가할수록 부하가 기하급수적으로 증가한다. 또한, PBFT를 개선하고자 한 선행연구는 블록체인의 트릴레마인 탈중앙화, 확장성, 보안성을 동시에 해결할 수 없다는 한계가 있다.

본 연구에서는 PBFT의 확장성 향상을 위한 분산 환경에서의 결함 노드 판단 및 대응 방법에 대해 제안하였다. 제안 방법은 PBFT 합의 과정에서 발생하는 메시지를 수집하여 노드의 신뢰도를 판단하고, 신뢰도에 따라 소규모 위원회를 구성하여 합의를 진행하였다. 우선순위에 기반한 선별적 합의 참여 기회를 제공하여 확장성을 개선하고, 결함 노드로 인한 지연을 막아 가용성을 향상시킬 수 있다. 또한, 분산 환경에서 별도의 대표자를 선출하지 않고 위원회를 구성하여 탈중앙화의 특성도 보존할 수 있다. 본 논문에서는 PBFT와 제안 방법의 지연시간, 처리량, 합의 성공률을 평가하였다. 평가 결과에 따르면, 제안 방법은 시간에 따른 인과 관계가 있는 결함 환경에서 결함에 효과적으로 대응하였다. 또한, 결함이 없는 환경에서는 제안 방법보다 PBFT가 효과적이지만, 결함 있는 환경에서 지연시간을 0.026s, 처리량을 833Mbps 빠르게 개선하였으며, 합의 성공률도 향상됨을 확인하였다.

목 차

논문 개요

Ⅰ. 서론 ····· 1

Ⅱ. 관련 연구 ····· 4

1. PBFT(Practical Byzantine Fault Tolerance) ····· 4

1) PBFT 개요 ····· 4

2) PBFT 동작 방식 ····· 5

2. PBFT 개선 연구 ····· 9

1) 대표 위원회 ····· 9

2) 다중 계층 ····· 11

3) 점수 ····· 13

4) 분할 ····· 16

Ⅲ. 분산형 결함 정보 판단 및 대응 방안 ····· 19

1. 결함 정보 ····· 22

2. 결함 노드 판단 ····· 24

3. 합의 위원회 구성 ····· 25

Ⅳ. 평가 ····· 27

1. 실험 환경 ····· 27

1) 신뢰성 모델 ····· 27

2) 평가 지표 ····· 29

3) 평가 프레임워크 ····· 30
2. 평가 결과 ····· 32
1) 신뢰성 모델별 평가 결과 ····· 32
2) 결함 유무에 따른 평가 결과 ····· 36

Ⅴ. 결론 ····· 39

ACKNOWLEDGEMENTS
참고문헌
ABSTRACT

표 차 례

Table I. Comparison of consensus algorithms ····· 5
Table II. Research on leader committee ····· 10
Table III. Research on multi-layer ····· 12
Table IV. Research on score ····· 14
Table V. Research on division ····· 16
Table VI. Experiment environment ····· 27
Table VII. Consensus success rate in fault environments ····· 36

그림 차례

FIGURE 1. Operation of practical byzantine fault tolerance ····· 6
FIGURE 2. Architecture of proposed method ····· 20
FIGURE 3. Flow chart of proposed method ····· 21
FIGURE 4. Sharing fault data from adjacency nodes ····· 23
FIGURE 5. Fault data structure ····· 24
FIGURE 6. Committee of proposed method ····· 26
FIGURE 7. Reliability model (infant mortality, random failures, wear out, bathtub curve) ····· 28
FIGURE 8. Class diagram of evaluation framework ····· 31
FIGURE 9. Latency and consensus rate of PBFT and proposed method by reliability model ····· 34
FIGURE 10. Throughput and consensus rate of PBFT and proposed method by reliability model ····· 35
FIGURE 11. Latency of PBFT and proposed method according to fault environments ····· 37
FIGURE 12. Throughput of PBFT and proposed method according to fault environments ····· 38

Ⅰ. 서 론

최근 4차 산업 혁명의 핵심 기반 기술인 블록체인 기술은 금융, 유통, 제조, 공공, 의료 등 산업과 사회의 전 분야에 걸쳐 활용범위를 확장해 나가고 있다[1]. 블록체인은 P2P 환경에서 신뢰할 수 있는 중개 기관 없이 트랜잭션을 기록하고 데이터를 분산하여 저장하는 장부 시스템이다[2]. 블록체인의 노드들은 정해진 규칙에 따라 거래 데이터 집합인 블록을 생성하고 합의한다. 블록은 시간 순서대로 연결되어 블록체인을 형성한다. 노드는 블록체인의 사본을 소유하여 데이터를 투명하게 관리할 수 있으며, 블록의 연결성은 데이터가 한 번 기록되면 변경하기 어렵게 만든다.

기존 중앙집중식 시스템은 중앙 기관이 신뢰를 보장하여 모든 구성원이 동일한 장부를 관리할 수 있다. 하지만 시스템을 유지·보수하기 위한 신뢰 비용을 중앙 기관에 지불해야 하며, 데이터를 한곳에 모아 관리하기 때문에 단일 장애점(Single point of failure) 문제가 발생한다. 분산형 시스템은 중앙 기관 없이 네트워크 참여자들이 공동으로 거래정보를 기록하고 관리할 수 있다. 모든 노드는 장부의 사본을 지니기 때문에 한 노드가 중단되더라도 다른 노드로부터 데이터를 공유받을 수 있다. 하지만 분산형 시스템에서 노드는 메시지의 지연, 훼손, 손실 등의 비잔틴 장애(Byzantine Fault)로 인해 상대방이 가진 장부를 신뢰할 수 없다[3]. 블록체인은 네트워크에 참여한 모두가 통일된 의사결정을 하고 신뢰할 수 있는 장부를 공유할 수 있도록 합의 알고리즘을 사용한다.

PBFT(Practical Byzantine Fault Tolerance) 합의 알고리즘은 비동기 네트워크에서 악의적이거나 비의도적으로 시스템 내부에 장애를 일으키는 노드가 존재하더라도 일정 수준의 장애를 허용하여 합의가 안정적으로 이루어질 수 있도록 한다[4]. PBFT는 전체 노드의 수가 $3f+1$ 일 때, 최대 f

개의 장애 노드가 있는 네트워크의 합의를 신뢰할 수 있음을 보장한다. PBFT는 request, pre-prepare, prepare, commit, replay 순으로 진행되며, 다섯 단계의 합의 과정을 통해 모든 노드가 동일한 장부를 유지할 수 있다. PBFT는 다른 합의 알고리즘에 비해 성능이 좋고, 합의가 완료되는 즉시 결과를 확정할 수 있어 활용도가 높다. 하지만 전체 노드의 $\frac{1}{3}$이상에 결함이 발생한다면 합의가 교착될 가능성이 크다. 또한, 다섯 단계의 합의 과정 중 prepare과 commit 단계에서 다량의 메시지가 발생하기 때문에 네트워크의 규모가 커질수록 복잡도가 급격하게 상승하여 확장성을 저해한다[5]. 기존의 연구들은 크게 네 가지 방법을 사용하여 PBFT의 한계점을 해결하기 위해 노력하였다. 하지만 PBFT 알고리즘의 확장성에 집중하여 문제를 해결하고자 하였기 때문에 확장성(Scalability), 보안성(Security), 탈중앙화(Decentralization)의 트릴레마를 해소하기 어렵다[6].

본 논문에서는 PBFT의 확장성 향상을 위한 분산 환경에서의 결함 노드 판단 및 대응 방법을 제안하고자 한다. PBFT 합의 과정에서 발생하는 메시지를 수집하여 노드의 신뢰도를 판단하고, 분산 환경에서 별도의 대표자를 선출하지 않고 소규모 위원회를 구성하여 합의를 수행한다.

제안 방법의 기여점은 다음과 같다.

- PBFT의 탈중앙화, 확장성, 보안성을 동시에 개선할 수 있는 방안을 제안한다. 우선순위에 기반한 선별적 합의 참여 기회를 제공하여 메시지 복잡도를 개선하고, 결함 노드로 인한 지연을 막아 가용성을 향상시켰다.
- 결함이 있는 환경과 없는 환경에서 PBFT와 제안 방법의 지연시간, 처리량, 합의 성공률을 평가하였으며, 제안 방법이 결함이 있는 환경에서 효과적으로 동작함을 확인하였다.

본 논문은 다음과 같이 구성된다. 논문의 2장에서는 PBFT에 대해 설명

하고, PBFT를 개선하고자 한 선행연구를 분석한다. 3장에서는 분산 환경에서 제삼자의 신뢰 보장 없이 결함 노드를 판단하고 대응하는 방법을 설명한다. 4장에서는 PBFT와 제안 방법을 지연시간, 처리량, 합의 성공률의 측면에서 평가하며, 5장에서 결론을 맺으며 마무리한다.

Ⅱ. 관련 연구

1. PBFT(Practical Byzantine Fault Tolerance)

1) PBFT 개요

PBFT는 비동기식 분산 네트워크에서 비잔틴 장애에 내성이 있는 합의 알고리즘이다[2]. 비잔틴 장애는 분산 네트워크 환경에서 발생할 수 있는 모든 형태의 장애를 의미한다[3]. 비잔틴 장애가 있음에도 불구하고 서비스가 정상적으로 동작하게끔 하는 BFT 계열의 대표적인 합의 알고리즘 중 하나가 PBFT이다. PBFT는 전체 노드가 $n=3f+1$개일 때, 최대 f개의 결함 노드를 견딜 수 있음을 증명하였다. PBFT를 사용해 분산 네트워크 환경에서 신뢰할 수 있는 제삼자 없이 하나의 의사결정을 내릴 수 있다.

FLP Impossibility 문제는 비동기 네트워크에서 Liveness와 Safty를 동시에 보장할 수 없음을 증명한 이론이다[7]. Liveness는 합의 대상과 노드에 문제가 없다면 합의를 달성할 수 있음을 의미하고, Safty는 합의를 달성했다면 모든 노드는 동일한 값에 접근할 수 있음을 의미한다. PBFT는 Livenes보다 Safty를 우선시하여 FLP Impossibility를 해결하고자 하였다. 따라서 PBFT는 메시지 지연시간이 무한히 증가하지 않는다는 가정하에 전체 네트워크의 합의를 달성할 수 있다.

표 1은 나카모토 계열 알고리즘인 PoW(Proof of Work), PoS(Proof of Stake)와 PBFT의 내결함성(Fault tolerence), TPS(Transaction per second), 최종성(Finality), 확장성(Scalability), 에너지 소비량(Power consumption)을 비교한 표이다[8, 9, 10]. PBFT는 타 알고리즘 대비 높은 TPS, 절대적 최종성, 낮은 에너지 소비량을 보장할 수 있다. 하지만 내결함성과 확장성 측면에서는 타 알고리즘 대비 낮은 성능을 보인다. 특히 PBFT 합의 과정에서 발생하는 메시지 복잡도는 $O(n^2)$으로 네트워크의 규모가 커

질수록 복잡도가 급격하게 상승하여 네트워크 규모를 확장하기 어려운 문제가 발생한다[5].

TABLE I
Comparison of consensus algorithms

Consensus algorithm	Fault tolerance	TPS	Finality	Scalability	Power consumption
PoW	50%	<100	probabilistic finality	strong	high
PoS	50%	<1000	probabilistic finality	strong	low
PBFT	33%	<2000	absolute finality	weak	low

2) PBFT 동작 방식

PBFT 네트워크의 참여자는 클라이언트(Client), 레플리카(Replica), 프라이머리(Primary) 노드, 백업(Back up) 노드로 나눌 수 있다. 클라이언트는 처음 합의 시작을 요청하는 노드이다. 레플리카는 원장의 복제본을 지닌 합의 참여자이다. 레플리카 중 하나는 프라이머리 노드로 지정되고, 나머지는 백업 노드의 역할을 수행한다.

PBFT는 총 다섯 단계로 합의를 수행한다.

1. Request : 클라이언트가 프라이머리 노드에게 request 메시지를 전송
2. Pre-prepare : 프라이머리 노드가 백업 노드에게 pre-prepare 메시지를 멀티캐스트
3. Prepare : 백업 노드는 prepare 메시지를 다른 노드에게 멀티캐스트
4. Commit : 레플리카 노드는 commit 메시지를 다른 노드에게 멀티캐스트

5. Reply : 레플리카 노드는 reply 메시지를 클라이언트에게 전송

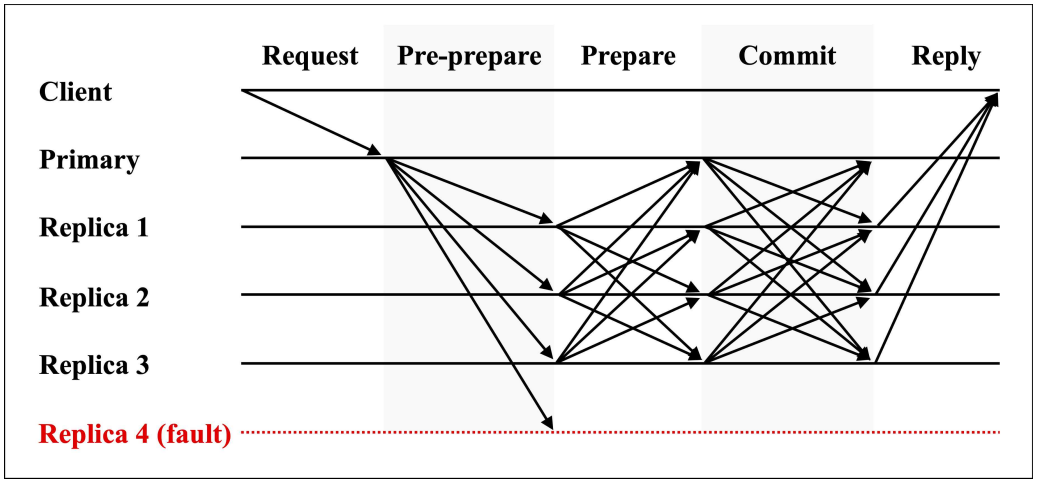

FIGURE 1. Operation of practical byzantine fault tolerance

① Request

Request 단계에서는 클라이언트가 프라이머리 노드에게 request 메시지를 전송한다. 클라이언트는 수식 1을 사용해 프라이머리 노드를 지정할 수 있다.

$$p = v \bmod n \qquad (1)$$

프라이머리 노드(p)는 뷰 번호(v)와 전체 레플리카 수(n)를 모듈로 연산하여 구할 수 있다. 예를 들어 뷰 번호가 12, 전체 레플리카 수가 10이라면 2번 노드가 프라이머리 노드로 지정된다.

클라이언트는 프라이머리 노드에게 request 메시지 $< REQUEST, o, t, c > \sigma_c$를 전송한다. o는 합의할 동작, t는 타임스탬프, c는 클라이언트 정보, σ_c는

클라이언트의 시그니처를 의미한다. 타임스탬프는 클라이언트가 메시지를 보낸 시간이 삽입되며, 이전에 발행된 request 메시지의 타임스탬프보다 높은 값이어야 한다. 프라이머리 노드는 클라이언트의 request 메시지를 수신한 뒤 pre-prepare 단계를 시작한다.

② Pre-prepare

Pre-prepare 단계에서는 프라이머리 노드가 백업 노드에게 멀티캐스트를 진행한다. 클라이언트로부터 request 메시지를 받은 프라이머리 노드는 현재 진행 중인 request의 수가 지정된 최댓값을 초과하지 않는 이상 즉시 프로토콜을 시작한다. 만약 최댓값을 초과한다면 노드의 버퍼에 메시지를 저장해두고, 우선 다른 메시지를 처리한 후에 순차적으로 request를 처리한다.

프라이머리 노드는 pre-prepare 메시지 $\ll PRE-PREPARE, v, n, d > \sigma_{p,} m >$ 를 프라이머리 노드의 로그로 기록한 뒤에 백업 노드에게 전송한다. 뷰 번호(v)는 합의의 단위를 의미하고, 연속적인 숫자가 할당된다. 시퀀스 번호(n)는 합의 진행 중인 블록의 번호로, 현재 블록 높이+1이다. 다이제스트(d)는 클라이언트가 전송한 request 메시지의 해시값이다. 백업 노드는 pre-prepare 메시지를 검토한다. 메시지를 수락하면 prepare 단계로 넘어가고, 메시지를 거절하면 아무런 동작도 하지 않는다.

③ Prepare

Prepare 단계에서는 백업 노드가 prepare 메시지를 멀티캐스트한다. 백업 노드는 prepare 메시지를 로그로 기록한 뒤, 자신을 제외한 모든 노드에게 prepare 메시지 $< PREPARE, v, n, d, i > \sigma_i$ 를 전송한다. i는 노드를 의미한다. prepare 메시지를 받은 노드들은 메시지를 검증하기 위해 시그니처 σ_i, 뷰 번호 v, 시퀀스 번호 n를 확인한다. 노드가 pre-prepare 메시지 하나와

prepare 메시지 $2f$개를 수락하면 prepared 상태가 되고, commit 단계를 시작한다.

④ Commit

Commit 단계에서는 모든 레플리카 노드들이 commit 메시지를 다른 레플리카 노드에게 멀티캐스트한다. 레플리카 노드 i는 commit 메시지를 로그로 기록한 뒤, 자신을 제외한 모든 노드에게 commit 메시지 $< COMMIT, v, n, D(m), i > \sigma_i$ 를 전송한다. $f+1$개의 결함 없는 레플리카가 prepared 상태가 되면 committed 상태가 된다. prepared 상태이고, $2f+1$개의 commit 메시지를 수락한다면 committed-local 상태가 된다. committed-local 상태의 노드는 reply 단계를 실시한다.

⑤ Reply

Reply 단계는 레플리카들이 reply 메시지를 클라이언트에게 전송한다. 레플리카들은 클라이언트가 보낸 동작(o)를 실행하고, 그 결과를 reply 메시지 $< REPLY, v, t, c, i, r > \sigma_i$ 에 담아 클라이언트에게 전송한다.

클라이언트는 레플리카로부터 $f+1$개의 동일한 메시지를 받으면 결과를 수용한다. 만약 레플리카로부터 충분한 메시지를 받지 못하였다면 모든 레플리카 노드에게 request 메시지를 브로드캐스팅한다. 레플리카 노드는 해당 메시지가 이미 처리되었다면 reply 메시지를 재전송하고, 처리되지 않았다면 프라이머리 노드에게 릴레이한다. 만약 프라이머리 노드가 request 메시지를 멀티캐스트하지 않으면 레플리카에 수용 가능 이상의 결함이 있는 것으로 의심하며 view change가 발생할 수 있다.

2. PBFT 개선 연구

본 장에서는 PBFT 개선을 위한 연구들을 분석하였다.

PBFT 개선 연구는 탈중앙화, 확장성, 보안성의 트릴레마를 직면한다[6]. 확장성은 네트워크의 규모가 증가하더라도 네트워크를 안정적으로 운용하는 것, 보안성은 모든 네트워크 참여자가 신뢰할 수 있는 동일한 장부를 공유하는 것, 탈중앙화는 네트워크가 중앙 집중화되지 않고 분산되어 운영되는 것을 의미한다. 하지만 선행연구들은 트릴레마의 일부만을 개선하였으며, 대부분 확장성을 개선하기 위해 탈중앙화와 보안성을 포기한다. 탈중앙화가 훼손되면 데이터의 신뢰성이 낮아지고, 확장성이 낮아지면 대규모 서비스에 적용하기 어려우며, 보안성이 낮아지면 서비스의 가용성을 보장하기 어렵다. 따라서 탈중앙화, 확장성, 보안성을 동시에 개선할 수 있는 연구가 필요하다.

최근 5년간의 PBFT 선행연구를 분석한 결과, 대표 위원회(Leader committee), 다중 계층(Multi-layer), 점수(Score), 분할(Division) 네 가지 방식으로 분류할 수 있었다. 대표 위원회 방식은 대표 노드를 선출하고, 일정 기간 동안 합의 권한을 대표 노드에게 일임한다. 다중 계층 방식은 기능 또는 역할에 따라 계층을 분리한다. 점수 방식은 노드의 신뢰도를 산정하고 일정 기준에 따라 노드를 처리한다. 분할 방식은 노드 또는 데이터를 분산하여 합의 부담을 감소시킨다. 선행연구들은 네 가지 방법을 단독 또는 복합적으로 사용하여 PBFT의 성능을 개선하고자 하였다. 아래에서는 PBFT 개선 선행연구를 네 가지 분류 기준에 따라 상세히 분석하였다.

1) 대표 위원회

대표 위원회 방식은 대표 노드를 선출하여 합의 권한을 위임한다. 합의에 참여하는 노드 중 합의를 주도할 대표 노드를 선출하거나, 전체 노드를 여

러 그룹으로 나누어 그룹을 대표할 노드를 선출하여 대표 위원회를 구성한다. 전체 노드를 대표하는 노드들만 합의에 참여하기 때문에 합의에 참여하는 노드의 수가 감소하여 네트워크 확장에 따른 성능 저하 문제를 개선할 수 있다. 표 2에서 대표 위원회 방식을 제안한 선행연구를 확인할 수 있다.

TABLE Ⅱ
Research on leader committee

Ref.	Key Technologies	Limitations
[11, 12]	• 의사 난수 생성기를 사용해 대표 노드 선출 • 블록별로 대표 노드를 변경	• 대표 노드 및 대표 위원회의 신뢰성을 보장하기 어려움 • 권한이 중앙 집중화될 우려 • 대표 노드가 사전에 노출 시, 공격에 취약해짐
[13]	• 다수의 소규모 위원회를 구성하여 내부에서 합의 진행 • 소규모 위원회에서 하위 블록을 병렬적으로 생성 • 위원회별 대표 노드를 선출하여 최종 블록 합의	
[14, 15]	• 신뢰할 수 있는 노드의 수에 따라 루트 위원회(root committee) 구성 • 루트 위원회를 신뢰할 수 없는 경우, 루트 위원회 전체를 교체	

Gosig와 LinBFT(Linear-Communication Byzantine Fault Tolerance)는 VRF(Verifiable Random Function)라는 의사 난수 생성기를 사용하여 대표 노드를 무작위하고 예측 불가능하게 선출하고, 블록별로 대표 노드를 변경하였다[11, 12].

ISCP(Improved Stellar Consensus Protocol)는 SCP(Stellar Consensus Protocol)의 한계점을 해결하기 위해 다수의 위원회를 구성하여 위원회 내부에서 합의를 진행하는 프로토콜 제안하였다[13]. SCP는 Pow와 BFT를 결합한 하이브리드형 합의 알고리즘으로 프로토콜의 효율성을 유지하면서도 최종 위원회의 정확성을 보장하기 어렵다는 한계가 있다. ISCP는 PoW의 연산 결과에 따라 노드를 여러 소위원회로 무작위하게 분할하여 하위 블록을 병렬적으로 생성하고, 소위원회 내부적으로 대표 노드를 선출하여 최종 블록을 합의하기 때문에 연산 복잡도와 오류 발생 가능성을 크게 감소시킬 수 있다.

Proteus와 Consistent BFT Protocol은 대규모 네트워크에서도 안정적인 성능을 보장하기 위해 전체 노드 중 신뢰할 수 있는 노드 수에 따라 루트 위원회(root committee)를 구성하여 합의를 진행하였다[14, 15]. 만약 루트 위원회가 합의를 달성하지 못하면, view change를 통해 루트 위원회 전체를 교체하여 보안성을 높일 수 있다.

대표 위원회 방식은 대표자를 선출하거나 대표 위원회를 구성하여 전체 네트워크 규모보다 작은 규모로 합의를 수행할 수 있다. 그러나 기존 연구에서는 대표자와 대표 위원회를 임의로 지정하기 때문에 대표자의 신뢰를 보장하기 어렵다. 대표자에게 권한이 집중되어 있으므로 대표자에게 결함이 발생할 경우 치명적인 결과를 초래할 수 있어 신뢰도 높은 대표자를 선출하는 것이 중요하다. 또한, 합의를 수행하기 전에 리더가 사전 노출되는 경우, 공격자의 타겟이 될 수 있다.

2) 다중 계층

다중 계층 방식은 기능 또는 역할에 따라 합의에 참여하는 노드를 그룹화하여 합의 계층을 분리한다. 계층별로 분리하여 저장하고 병렬적으로 처리

하여 노드가 확장함에 따라 발생하는 시간 지연과 성능 저하 문제를 개선하고자 하였다. 표 3에서 다중 계층 방식의 선행연구를 확인할 수 있다.

TABLE Ⅲ
Research on multi-layer

Ref.	Key Technologies	Limitations
[16]	• 장치의 기능에 맞추어 Infrastructure, Core, High-Level 계층으로 분리한 3계층 구조로 합의 수행	• High-Level 계층 노드로 권한이 중앙 집중화 • 공개된 노드를 보호하기 위한 대책이 필요함
[17]	• 거래 검증(참여 노드를 검증함), 합의 형성(노드를 식별하여 지정된 규칙에 따라 행동) 두 단계로 합의 수행	• 참여자 검증을 위한 추가적인 비용이 소모됨
[18]	• 하위 계층을 여러 개의 소규모 그룹으로 나누어 합의를 수행하는 x-layer 제시	• 첫 번째 계층의 규모에 제한이 있어 악성 노드가 ⅓ 이상 구성될 가능성이 높음 • 계층이 중첩될수록 그룹의 규모가 작아져 취약성이 증가함

Rashid 등이 연구한 보안 프레임워크는 Infrastructure, Core, High-Level 계층으로 나누어 합의를 진행한다[16]. Infrastructure 계층은 IoT(Internet of Things) 노드, Core 계층은 Cluster head(CH), 동기화 노드 등 제어기기, High-Level 계층은 기지국으로 구성되며 각 장치에 맞는 능력 및 기능에 따른 역할을 수행한다. 하지만 High-Level 계층 노드로 권한이 중앙 집중화될 우려가 있으며, 공개된 노드를 공격자로부터 보호하기 위한 대책이 필요하다.

PoBT(Proof of Block and Trade)는 거래 검증과 합의 형성 두 단계로 분리하여 합의한다[17]. 거래 검증 단계에서는 스마트 계약을 통해 검증된 노드만 합의에 참여하도록 허가하여 보안을 유지하면서도 오버헤드를 제어할 수 있다. 합의 형성 단계에서는 참여 노드의 서명, 체인 코드의 유효성, 주문자 고유 식별값이 유효한 것으로 확인되면 수신된 임의 번호를 반환하여 지정한 규칙에 따라 제한된 노드만 합의에 참여할 수 있도록 하여 확장성을 높였다. 하지만 참여자 검증을 위한 추가적인 비용이 소모되는 단점이 있다.

Li 등이 제시한 다중 계층 PBFT 합의 알고리즘은 네트워크 구성원을 여러 집합과 계층으로 나누어 합의를 수행한다[18]. 하위 계층을 여러 개의 소규모 그룹으로 나누어 메시지 복잡도를 낮출 수 있다. 하지만 첫 번째 계층의 수가 제한되어 있고, 계층이 중첩될수록 그룹의 규모가 작아지기 때문에 악성 노드에 장악되기 쉽다.

다중 계층 방식은 합의 과정에서 발생하는 노드의 부담을 분산하여 PBFT의 확장성을 보장할 수 있다. 하지만 합의 노드의 규모가 줄어들어 보안성을 보장하기 어렵고, 추가적인 비용이 소모되는 한계가 있다.

3) 점수

점수 방식은 합의에 참여하는 노드의 신뢰값을 계산하고 지정된 점수 메커니즘에 따라 노드를 처리한다. 계산된 신뢰값을 임계값과 비교하여 해당 노드를 신뢰할 수 있는지 판단한다. 신뢰도에 따라 다양한 방법으로 노드를 처리하여 성능과 보안성을 향상시킬 수 있다. 신뢰도에 따라 노드를 그룹별로 분리하여 신뢰할 수 있는 노드로 판단한 그룹에는 인센티브를 제공하여 정상 행위를 지속하도록 유도할 수 있다. 또한, 신뢰도가 가장 낮은 노드를 폐기하고 합의를 진행할 수 있다. 표 4에서 점수 방식의 선행연구를 확인할 수 있다.

TABLE Ⅳ

Research on score

Ref.	Key Technologies	Limitations
[3]	• EigenTrust 모델 기반으로 노드의 신뢰도 측정 • Multi-layer 기반 노드별 trust value 값 계산하여 악성 노드 감지	• 노드의 신뢰도를 회복할 수 있는 추가적인 방안이 필요함
[19]	• P2P trust calculation 모델 기반으로 노드의 신뢰도 측정	
[20]	• Reputation capital 기반 평판 모델 구축 및 IoT 기기의 신뢰도 측정 • 그룹별 필요한 RC 점수 정의 후 해당 점수에 부합하는 노드 분류	
[21]	• 노드의 상태, 처리 지연, 전달 속도 및 응답 시간을 바탕으로 최종 노드 통신 품질을 계산. 해당 값이 임계값을 넘어가면 악성 노드라 판단	• 정보를 블록체인에 저장하여 추가적인 비용이 필요함
[22]	• 판정 결과와 최종 결과와 일치하면 +1점, 그렇지 않으면 -5점 부여 • 점수 메커니즘에 따라 합의 노드 집합과 후보 노드 집합 분류	• 결함으로 인한 낮은 점수 획득 시에도 악성 노드로 판단되어 합의에 참여하지 못할 수 있음

[23]	• 신뢰도가 가장 낮은 비콘 노드를 폐기하여 WSN에서 로컬리제이션의 안정성과 일관성을 보장	• 신뢰도가 낮은 비콘 노드를 바로 폐기해 고의성 없는 결함 노드의 신뢰 회복이 불가능
[24]	• 신뢰도 순위 기반의 합의 프로토콜을 기반으로 경제적 이익과 잘못된 행동을 고려하는 인센티브 메커니즘을 제시	• 시스템의 보안성을 제고할 수 있으나, 확장성 문제는 해결하기 어려움

먼저, 신뢰값을 특정 임계값과 비교하여 그룹별로 분리하는 연구이다[3, 19, 20, 21, 22]. T-PBFT, Trust-PBFT, Reputation Capital(RC) 모델은 신뢰도 산정 모델을 사용하여 악성 노드를 판단하였다[3, 19, 20]. BTM(Blockchain Trust Model)은 노드의 상태, 처리 지연, 전달 속도 및 응답 시간을 바탕으로 신뢰값을 계산하고, 해당 값이 임계값을 초과할 때 악성이라고 분류하였다[21]. SG-PBFT는 악성의 유무를 점수로 부여하였는데, 모든 노드의 초기 점수를 100점으로 설정하고 임계값과 신뢰값이 일치하면 1점, 그렇지 않으면 -5점을 부여하여 합의 노드 집합과 후보 노드 집합을 분류하였다[22].

블록체인 기반 신뢰 관리 모델과 CertChain은 신뢰도가 낮은 노드를 합의에서 제외하는 연구이다[23, 24]. 신뢰값이 가장 낮은 비콘 노드를 폐기하고, 신뢰값이 가장 높은 노드에만 대표자 권한을 부여하여 합의 과정에서의 안정성과 일관성을 보장하였다. 그러나 신뢰도가 낮은 노드를 바로 폐기함으로써 고의성 없는 결함 노드들이 신뢰를 회복할 수 없다는 한계가 존재한다.

신뢰도가 좋은 노드에게 인센티브를 제공하는 연구는 시스템의 보안성을 제고할 수 있으나, 노드가 증가함에 따라 발생하는 지연 문제는 해결하기 어렵다. 또한, 신뢰도가 좋지 않은 노드에게 디파짓을 제공하는 연구는 낮은 신뢰

도를 지닌 노드의 성능이 좋아지더라도 합의에 참여하지 못하게 되는 문제가 발생할 수 있다.

4) 분할

분할 방식은 노드 또는 데이터를 분할하여 합의를 수행한다. 전체 노드를 복수의 클러스터로 나누어 클러스터별로 합의를 수행하거나, 데이터를 여러 개의 블록으로 분할하여 노드 전체에 분산한다. 표 5는 분할 방식의 선행연구를 정리하여 나타낸 표이다.

TABLE V
Research on division

Ref.	Key Technologies	Limitations
[16]	• EC (Evolutionary Computation)과 SI((Swarm Intelligence) 기반의 지능형 클러스터링 기법 보안 메커니즘	• 악성 노드가 클러스터의 다수를 차지할 가능성이 있음
[25]	• 전체 노드를 복수의 클러스터로 분할해 클러스터 단위별 합의 수행	• 선형 통신 방식과 프라이머리 노드 선정 방식에서 보안성이 약화됨 • 클러스터의 크기에 따라 합의 지연시간이 증가함
[26]	• 임계값으로 설정한 통신 긴밀도 수준에 따라 커뮤니티 분류	• 확장성 문제를 해결하는 방안이 되지만, 블록체인 트릴레마에 따른 일부 지연성과 보안성 문제 발생 • 임계값에 따른 분류로 악성이 아닌, 결함으로 인한 낮

		은 점수 획득 시 합의에 참여하지 못함 • 일부 노드는 상호 연결 및 운용성이 없음
[27]	• 샤딩 기반의 컴퓨팅, 저장 및 전송능력을 고려해 IoT 노드를 그룹화하는 경량 블록체인	• 악성 노드가 클러스터의 다수를 차지할 가능성이 있음
[28]	• 네트워크 코딩의 개념을 적용해 전체 데이터를 여 블록으로 나눠 노드에 분산	• 확장성 문제를 해결하는 방안이 되지만, 블록체인 트릴레마에 따른 일부 지연성과 보안성 문제 발생 • 데이터 블록 크기가 작을 경우에만 효율적 • 해시 함수로 생성된 메시지 요약을 전송하는 것이 더 효율적

먼저 전체 노드를 복수의 클러스터로 나누어 합의를 수행하는 연구이다[16, 25, 26]. Rashid 등이 연구한 보안 프레임워크는 GA(Genetic Algorithms)와 PSO(Particle Swarm Optimization)로 구성된 진화 계산 알고리즘을 사용하여 클러스터 단위로 노드를 분리한다[16]. 이때, 각 클러스터 헤더가(CH;Cluster Header) 권한을 가지고 합의에 참여한다. CBS-PBFT는 노드를 복수의 클러스터로 분리하여 CN(Client Node), CPN(Cluster Primary Node), LPN(Local Primary Node), BN(Backup Node)의 4가지 역할로 구분지었다[25]. 합의 과정에서 선형 통신 방식을 도입하고, 클러스터 내에서 합의 프로세스를 진행하며 블록체인의 확장성 문제를 개선하고자 했다. 그러나 선행 통신 방식과 CPN과 LPN의 프라이머리

노드를 선정하는 방식이 보안성을 약화시킨다. Suisheng Li가 제안한 프레임워크는 통신 긴밀도와 유사 기능에 따라 클러스터를 구성한다[26]. 이때, 통신 긴밀도는 단위 시간당 노드 사이의 통신량으로 정의한다. 클러스터 간 공통 노드를 구성하고 있어, 노드는 동시에 여러 클러스터에 포함될 수 있다. 이는 통신 긴밀도의 임계값에 따라 노드를 분류하고 있어 악성이 아닌, 결함으로 인해 낮은 점수를 획득할 경우 합의에 참여할 수 없다는 치명적인 문제가 발생한다. 또한, 노드의 일부는 상호 연결이나 운용성이 없어 특정 노드와만 통신하는 등 합의의 정확성을 보장할 수 없다. FastChain은 컴퓨팅, 저장 및 전송 기능에 따라 군사 애플리케이션, 군인 및 드론을 사용하여 네트워크 전투 시나리오를 시뮬레이션한 환경의 IoBT(Internet-of-Battlefield-Things) 노드를 그룹화하였다[27].

데이터를 분할하여 합의를 수행하는 연구인 Network-Coded PBFT는 샤딩(sharding) 프로토콜에 네트워크 코딩 개념을 추가하여 적은 대역폭으로 합의에 도달할 수 있는 방법을 제안하였다[28]. 전체 데이터를 여러 블록으로 나눠 노드 전체에 분산하며 노드 전체에 필요한 최대 대역폭을 줄였다. 사전에 정의된 규칙에 따라 일부 블록만 합의를 진행해 통신 복잡도를 유지하며 확장성이 향상된다. 그러나 블록체인에 빅데이터가 저장되는 경우, 높은 합의 지연을 발생시키기 때문에 비효율적이다. 또한, 블록의 크기가 작다는 가정하에 prepare, commit 단계에서 일반적인 데이터를 전송하지만, 해시값을 전송하는 경우보다 낮은 성능을 제공한다.

분할 방법은 확장성에 따른 합의 지연을 개선하고자 다수의 소그룹이나 블록을 분할하였지만, 클러스터 크기 및 데이터 크기가 확장됨에 따라 여전히 높은 합의 지연이 발생한다. 또한, 전체 노드 수가 동일한 환경에서는 클러스터 수보다 크기가 합의 지연에 영향을 주고 있어 확장성에 따른 합의 지연과 보안성 문제는 필연적이다.

Ⅲ. 분산형 결함 정보 판단 및 대응 방안

본 논문에서는 분산 환경에서 제삼자의 신뢰 보장 없이 결함 노드를 판단하고 대응하는 방법을 제안한다. 제안 방법은 노드의 이전 행위를 기반으로 우선순위를 산정한다. 추가적으로 결함 노드를 더욱 정확하게 판정하기 위하여 인접 노드로부터 결함 정보를 공유받을 수 있다. 각 노드는 산정한 우선순위를 토대로 상위 c개의 노드와 합의를 수행한다. 제안 방법은 신뢰도에 따라 선별적으로 합의 참여 기회를 부여함으로써 PBFT의 성능과 보안성을 향상시킬 수 있다.

그림 2는 제안 방법의 구조도이다. 제안 방법은 우선순위 산정, 결함 정보 공유, 위원회 기반 합의로 나누어 볼 수 있다.

우선순위 산정은 노드의 이전 행위를 기반으로 해당 노드가 결함이 발생할 확률이 높은지 예측하는 단계이다. 합의 과정에서 발생한 메시지를 집계하여 메시지를 전송한 노드를 기록한다. 이를 결함 정보(Fault data)로 부르며, 우선순위 산출을 위해 사용한다. 이때, 모든 합의 단계에 성실하게 메시지를 전송한 노드가 결함을 일으킬 가능성이 적다고 판단한다.

결함 정보 공유는 우선순위를 더 정확히 산출하기 위해 필요한 단계이다. 노드는 인접 노드에게 결함 정보를 요청하고, 인접 노드로부터 공유 받은 결함 정보를 추가하여 우선순위를 산출한다.

위원회 기반 합의는 노드가 위원회를 조직하여 합의를 수행하는 단계이다. 합의 위원회는 산출한 우선순위를 기반으로 상위 c개($c < n$)의 노드로 구성되며 각 노드는 자신의 경험에 따라 각기 다른 위원회를 조직하게 된다.

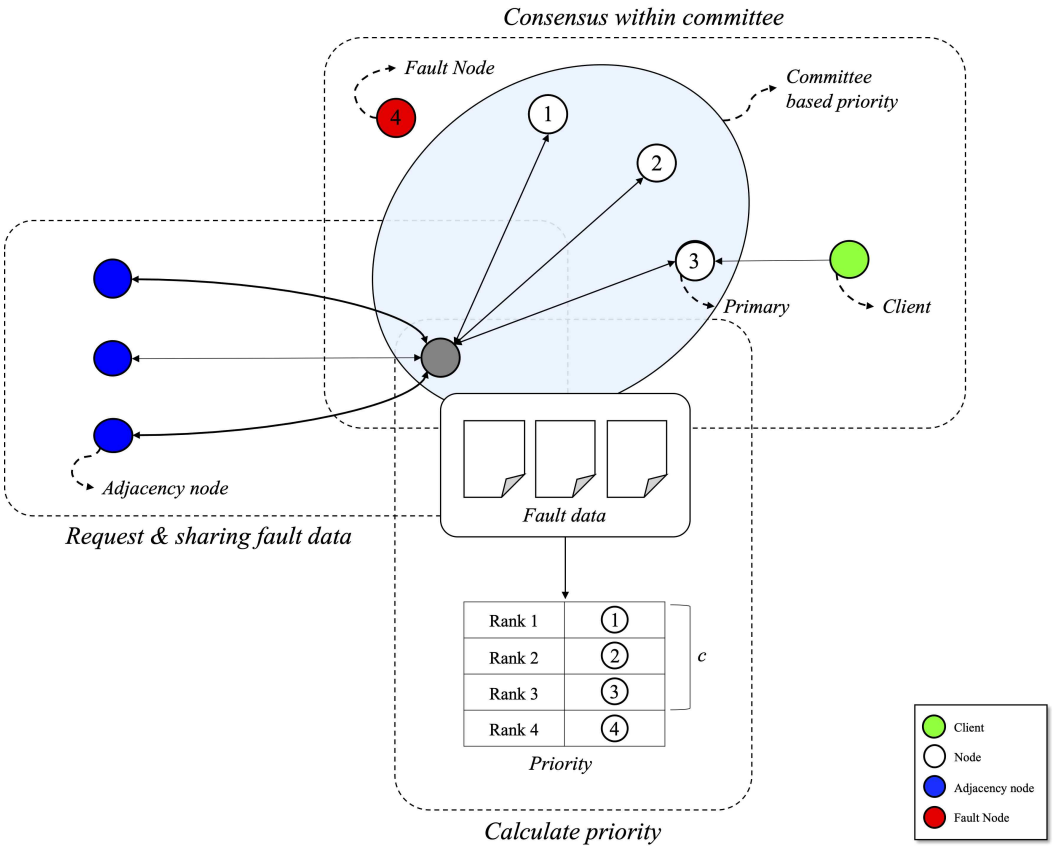

FIGURE 2. Architecture of proposed method

제안 방법은 세 가지 장점이 있다.

첫 번째로 메시지 복잡도가 감소한다. 제안 방법은 우선순위에 따라 결함 노드를 제외하고 합의를 수행한다. 따라서 전체 네트워크 규모보다 작은 규모로 합의를 수행할 수 있으므로 합의 과정에서 발생하는 높은 메시지 복잡도 문제를 해결할 수 있다.

두 번째로 보안성이 향상된다. 합의에 참여할 노드를 임의로 지정하는 것이 아니라 결함이 발생할 가능성이 높은 노드를 예측하여 제외하기 때문에 합의 위원회에 결함 노드가 포함될 가능성이 작아진다. 따라서 결함 노

드로 인한 지연이 발생할 가능성이 낮아지고, 합의 성공률이 높아져 네트워크의 가용성을 보장할 수 있다.

마지막으로 특정 노드에게 권한이 집중되지 않는다. 기존 연구처럼 대표 노드가 임의의 위원회를 구성하는 방식이 아니라 각 노드의 경험을 기반으로 각자의 위원회를 생성하기 때문에 위원회 구성을 전담할 노드가 필요하지 않다.

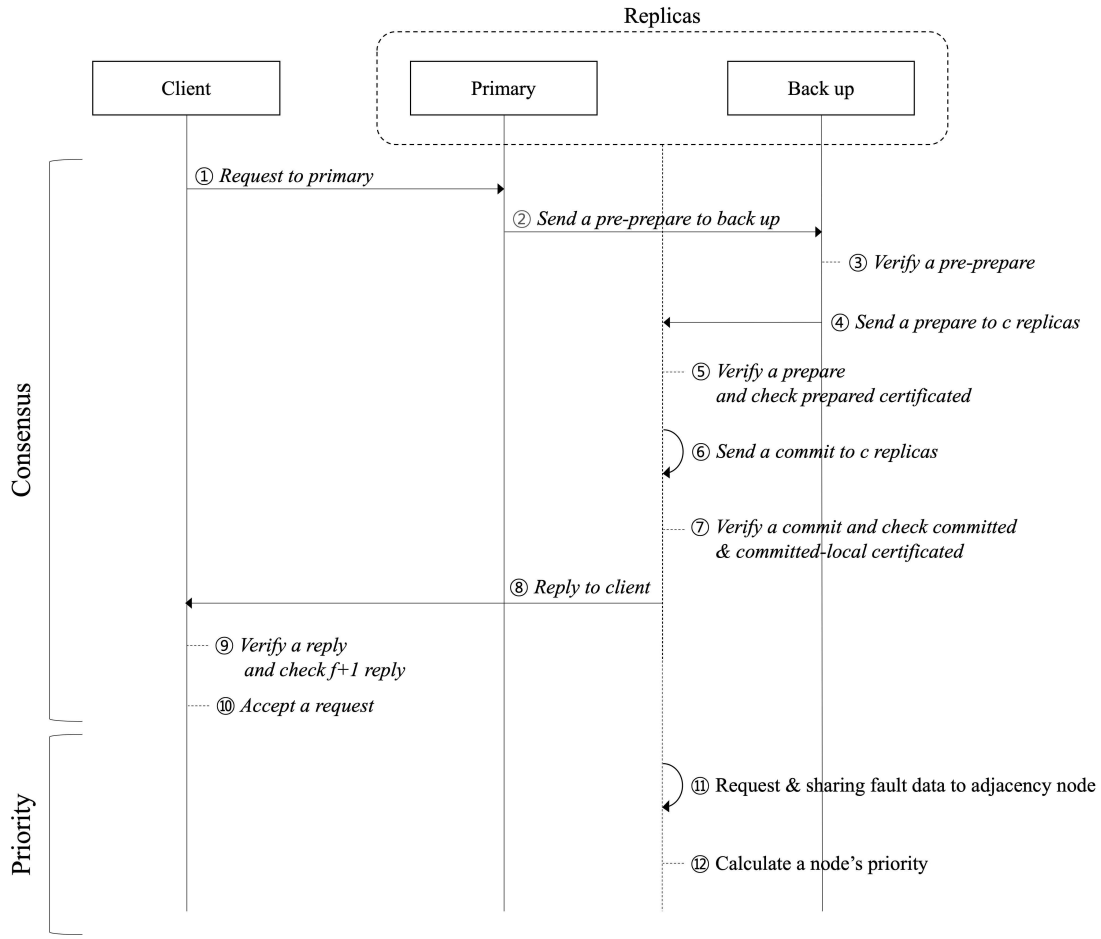

FIGURE 3. Flow chart of proposed method

그림 3은 구성원에 따른 제안 방법의 동작 방식을 확인할 수 있는 플로우차트이다. 크게 합의를 수행하는 합의(Consensus) 단계와 합의 완료 후 우선순위를 산출하는 우선순위(Priority) 단계로 나뉜다.

합의 단계는 PBFT 동작 방식과 유사하다. 처음 클라이언트가 프라이머리 노드에게 합의 요청을 보내며 시작한다. 프라이머리 노드는 request 메시지를 확인하고 pre-prepare 메시지를 백업 노드에게 전송한다. 백업 노드는 pre-prepare 메시지를 검증하고, 우선순위가 높은 c개의 노드에게 prepare 메시지를 전송한다. prepare 메시지를 받은 노드는 prepare 메시지를 검증하고, 다시 우선순위가 높은 c개의 노드에게 commit 메시지를 전송한다. 노드가 committed-local 상태가 되면 클라이언트에게 reply 메시지를 전송한다. 클라이언트는 reply 메시지를 $f+1$개 받으면 request를 수용한다.

우선순위 단계는 먼저 레플리카 노드가 인접 노드에게 결함 정보를 요청하고 공유받는다. 이후, 자신의 결함 정보와 공유받은 결함 정보를 토대로 우선순위를 산출한다. 산출한 우선순위는 다음 합의에서 사용한다.

1. 결함 정보

제안 방법은 PBFT 합의 과정에서 발생하는 메시지를 집계하여 결함 노드를 예측한다. PBFT는 prepare과 commit 단계에서 모든 노드가 모든 노드에게 합의 메시지를 전송한다. 합의 과정에서 노드는 결함 여부에 따라 메시지 전송에 성공할 수도 있고, 메시지 전송에 실패할 수도 있다. 전체 노드의 수가 n이고 결함 노드의 수가 f라면, 두 단계에서 발생하는 메시지는 총 $2(n-f)(n-f)$개이다.

각 노드는 합의 과정에서 발생한 모든 메시지를 기록한다. 기록된 메시지를 토대로 결함 정보를 생성한다. 결함 정보는 블록 번호(Block

number), 뷰 번호(View number), 폴트(Fault)로 구성된다. 블록 번호는 블록 순서에 관한 정보이다. 뷰 번호는 블록 생성에 실패하여 합의를 재시작한 횟수이다. 두 정보를 통해 어느 시기에 발생한 결함 정보인지 확인할 수 있다. 만약 두 번째 블록을 합의하기 위한 첫 번째 뷰라면 블록 번호는 2, 뷰 번호는 1로 기록된다. 폴트는 결함 노드에 관한 정보로, n(전체 노드 수)비트로 구성된 데이터 집합이다. 노드는 받은 메시지를 집계하여 prepare 단계와 commit 단계 모두 메시지를 성공적으로 전송한 노드 번호와 일치하는 위치에 1을 기록한다. 예를 들어 전체 5개의 노드 중 3, 5번 노드로부터 메시지를 받았다면 폴트는 [0, 0, 1, 0, 1]로 기록된다.

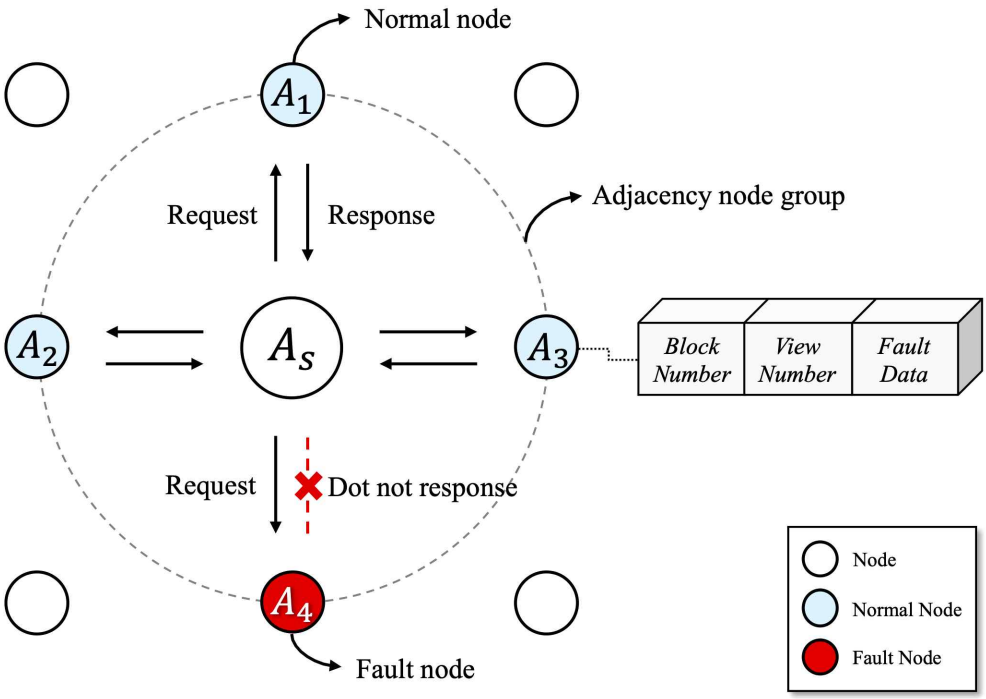

FIGURE 4. Sharing fault data from adjacency nodes

결함 노드를 더욱 정확하게 판단하기 위해 인접 노드로부터 결함 정보를 공유 받을 수 있다. 노드는 인접한 k개의 노드에게 결함 정보 공유를 요청한다. k개의 인접 노드에게 결함 정보를 요청하면 결함이 아닌 노드들로부터 결함 정보를 전달받을 수 있다. 만약 인접 노드에게 결함이 발생하여 결함 정보를 받지 못한다고 하더라도 다른 인접 노드에게 추가적으로 결함 정보를 요구하지 않는다. 폴트 데이터의 첫 번째 자리에 자기 자신의 결함 정보(A_s)를 저장하고, 순차적으로 인접 노드로부터 받은 결함 정보(A_1~A_k)를 저장한다.

Block number	*View number*	*Fault data*
2	1	[0, 1, 1, 0, 0], [0, 1, 1, 1, 0], [0, 1, 1, 0, 0]
3	1	[0, 1, 1, 0, 0], [0, 1, 1, 1, 0], [0, 1, 1, 0, 0]
3	2	[0, 1, 1, 0, 0], [0, 1, 1, 1, 0], [0, 1, 1, 0, 0]
4	1	[0, 1, 1, 0, 0], [0, 1, 1, 1, 0], [0, 1, 1, 0, 0]

A_s $A_1 \sim A_k$

FIGURE 5. Fault data structure

2. 결함 노드 판단

생성한 결함 정보를 토대로 우선순위를 산출하여 노드의 신뢰도를 판단한다. 우선순위 산정의 목적은 신뢰성 높은 노드를 결정하는 것이 아니라 결함 발생 가능성이 높은 노드를 최대한 합의에서 제외하는 것이다. 노드는 일정 기간 동안의 결함 정보를 합산하여 우선순위를 산출하고, 합산된 값이 클수록 노드의 신뢰성이 높다고 판단한다. 예를 들어 노드가 [0, 1, 0,

1, 0], [0, 1, 1, 1, 1], [0, 1, 0, 0, 1] 세 개의 결함 정보를 가지고 있다면, 합산한 값은 [0, 3, 1, 2, 2]이고 2, 4, 5, 3, 1 번 노드 순서대로 신뢰성이 높다고 판단할 수 있다. 기간은 임의로 설정할 수 있으며 오래된 데이터는 삭제하고 최근 데이터만을 반영하여 초반에 결함이 많이 발생했던 노드가 지속적으로 합의에 제외되지 않도록 한다.

각 노드들이 가진 결함 정보는 모두 다르기 때문에 노드의 경험에 따라 각기 다른 우선순위를 산출하게 된다. 결함 정보가 없어 우선순위를 산정할 수 없을 때에는 PBFT와 동일하게 동작한다.

3. 합의 위원회 구성

노드는 우선순위를 기반으로 합의 위원회를 구성한다. 각 노드는 산정한 우선순위를 토대로 신뢰도가 높은 c개의 노드에게 합의 메시지를 전송한다.

그림 6은 한 노드를 기준으로 분석한 그림이다. 노드 D를 기준으로 노드 D가 산정한 우선순위가 높은 c개의 노드에게는 메시지를 전송하고, 노드 D를 우선순위가 높은 c개의 노드로 선정한 노드로부터는 메시지를 받는다. 미리 구성한 위원회 내부에서 메시지 송수신이 이루어지는 기존 선행연구와는 달리, 제안 방법은 노드가 구성한 위원회 외부로부터 메시지를 전달받을 수 있다. 노드는 다른 노드로부터 $c/2+1$개의 메시지를 받을 시, reply 메시지를 클라이언트 노드에게 전송한다. 각 노드가 구성하는 합의 위원회는 총 $n-f$개가 생성된다. 합의 위원회 노드 수가 c, 합의 위원회로 선출된 결함 노드의 수가 b일 때, 메시지 복잡도는 $2(n-f)(n-f)$에서 $2(c-b)(c-b)$로 낮아진다.

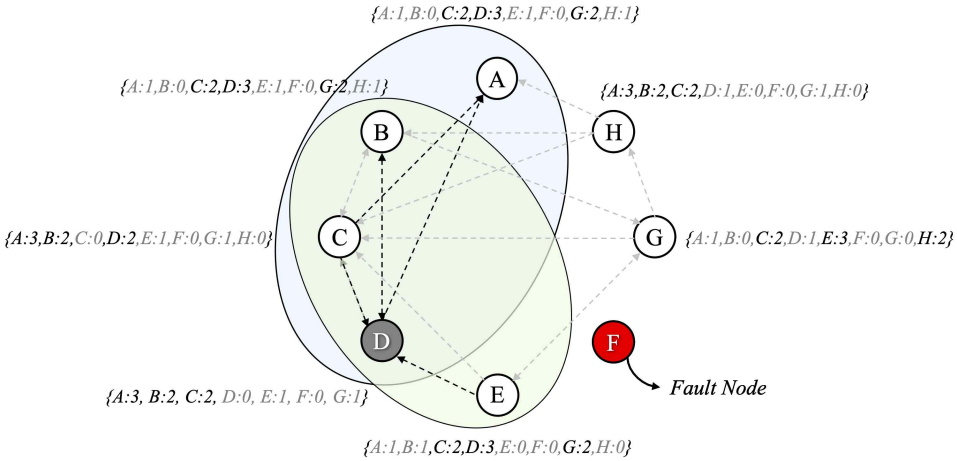

FIGURE 6. Committee of proposed method

Ⅳ. 평가

1. 실험 환경

4장에서는 블록체인 네트워크 환경을 단순하게 재현한 Python 시뮬레이터를 구현하여 PBFT와 제안 방법의 지연시간, 처리량, 합의 성공률을 평가하였다. 실험 환경은 표 6과 같다. 평가는 10-core CPU Apple M1 Pro chip, RAM 16GB, MacOS 12.0.1 환경에서 수행하였으며 시뮬레이터는 Python 3.10.1 버전으로 제작하였다.

TABLE Ⅵ
Experiment environment

구분	사양 및 버전
CPU	10-core CPU Apple M1 Pro chip
RAM	16GB
OS	MacOS 12.0.1
Python	3.10.1

1) 신뢰성 모델

신뢰성 모델(Reliability model)은 시간에 따른 장치의 수명을 예측하여 장치의 고장을 추정, 예방 및 관리할 수 있는 모델이다. 신뢰성 모델 중 노후화로 인한 결함을 반영할 수 있는 wear out과 무작위 결함을 반영할 수 있는 infant mortality, random failures를 사용하였다[29]. 시간의 흐름에 따라 변화하는 세 신뢰성 모델의 결함 확률은 그림 8에서 확인할 수 있다.

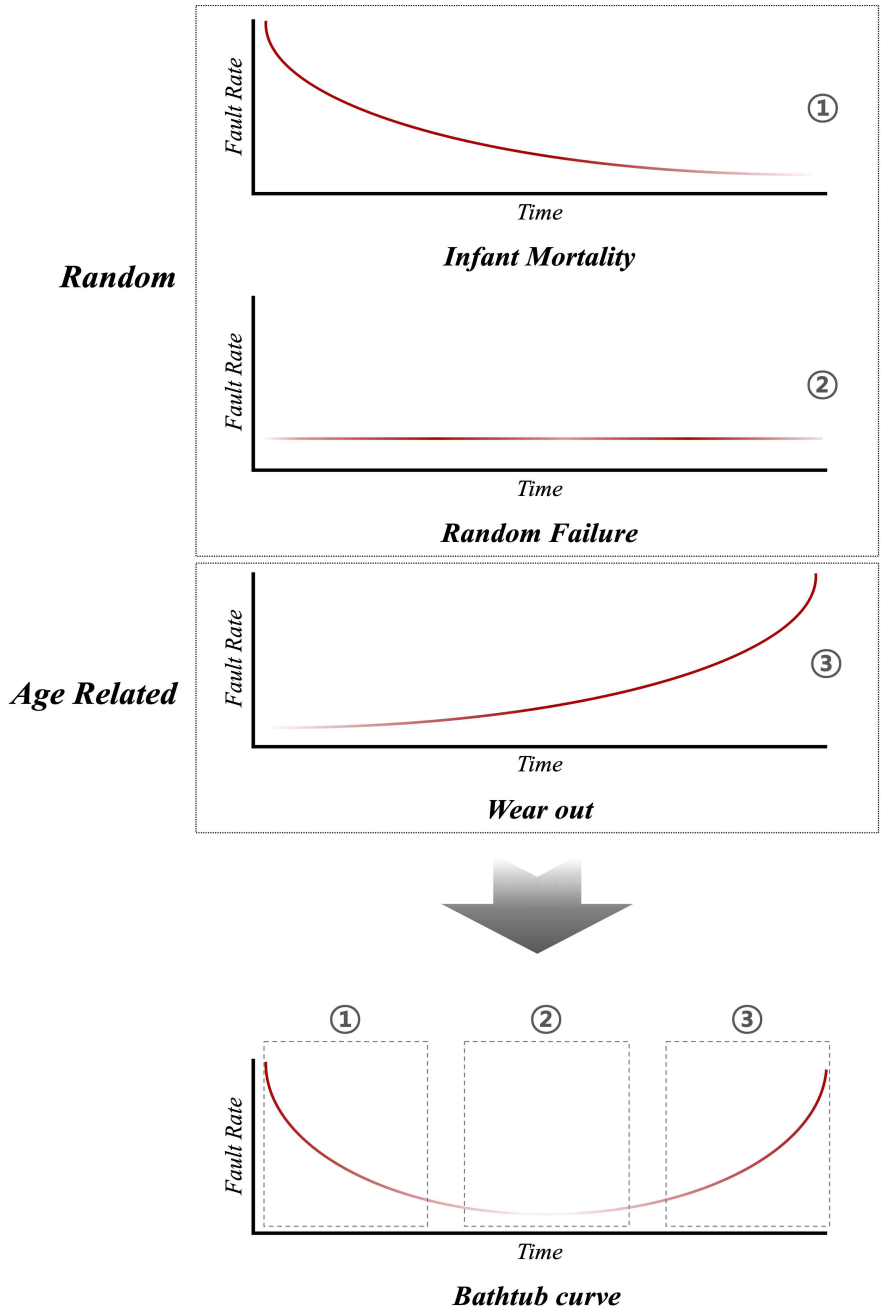

FIGURE 7. Reliability model (infant mortality, random failures, wear out, bathtub curve)

Infant mortality 모델은 장치나 소프트웨어 도입 초기에 발생할 수 있는 결함을 반영하며 초반에 결함 발생 확률이 높고 점차 안정화되는 형태를 보인다. Random failures 모델은 무작위로 발생하는 결함을 반영하며 시간과 상관없이 일정하고 낮은 결함 확률을 보인다. Wear out 모델은 장치의 노후화로 인한 결함을 반영하며 시간이 지날수록 결함 발생 확률이 높아진다.

신뢰성 모델은 신뢰성 모델과 관련한 다양한 확률, 통계 모델을 제공하는 파이썬의 Reliability 라이브러리를 사용하여 구현하였다. Infant mortality는 웨이불 분포(Weibull distribution), random failures는 지수 분포(Exponential Distribution), wear out은 로그 정규분포(Log-normal distribution)로 구현하였다. 신뢰성 모델별로 1부터 1000까지 시간의 흐름에 따른 결함 발생 확률을 생성하였다.

추가적으로 세 모델을 결합한 bathtub curve 모델을 사용하였다. Bathtub curve 모델의 초기는 infant mortality, 중기는 random failures, 말기는 wear out 모델의 경향성을 보인다. 최종적으로 bathtub curve 모델을 랜덤으로 슬라이싱하여 노드에게 부여함으로써 현실과 유사한 결함 모델을 반영하고자 하였다.

2) 평가 지표

제안 방법을 평가하기 위해 세 가지 평가 지표를 선정하였다.

첫 번째는 지연시간(Latency)이다. 지연시간은 합의를 시작할 때부터 종료할 때까지의 시간을 의미한다. PBFT의 지연시간은 클라이언트가 request 메시지를 전송하는 시점부터 reply 메시지를 받는 시점까지를 측정하였다. 제안 방법의 지연시간은 클라이언트가 request 메시지를 전송하는 시점부터 우선순위를 산출 완료하는 시점까지 측정하였다. 개별 노드의 통신 시간, 블록체인 동작 시간 등 합의 알고리즘과 관련 없는 항목은 측정에서 배제하였

다.

두 번째는 처리량(Throughput)이다. 처리량은 성공적으로 처리 완료한 유효 정보량을 의미한다. 처리량은 수식 2를 통해 측정할 수 있다. 현재 블록의 높이와 블록 사이즈를 곱하여 유효 정보량을 산출하고, 유효 정보량을 지연시간으로 나누어 초당 비트 전송 수를 측정하였다.

$$Throughput(bps) = \frac{(block\ height \times block\ size)}{latency} \qquad (2)$$

세 번째는 합의 성공률(Consensus success rate)이다. 합의 성공률은 전체 합의 시도 횟수 중 블록 합의에 성공한 비율을 의미한다. 합의 성공률은 수식 2를 통해 측정할 수 있다.

$$Consensus\ success\ rate(\%) = \frac{block\ height}{the\ total\ number\ of\ consensus} \qquad (3)$$

3) 평가 프레임워크

제안 방법을 평가하기 위해 블록체인의 동작 방식을 단순화하여 Python 기반 시뮬레이터로 구현하였다. 평가 프레임워크는 신뢰성 모델(Reliability), 노드(Node), 합의 알고리즘(Consensus Algorithm), 블록체인(Blockchain), 시뮬레이션(Simulation), 평가(Evaluation)로 구성된다. 그림 8은 평가 프레임워크의 클래스 다이어그램이다.

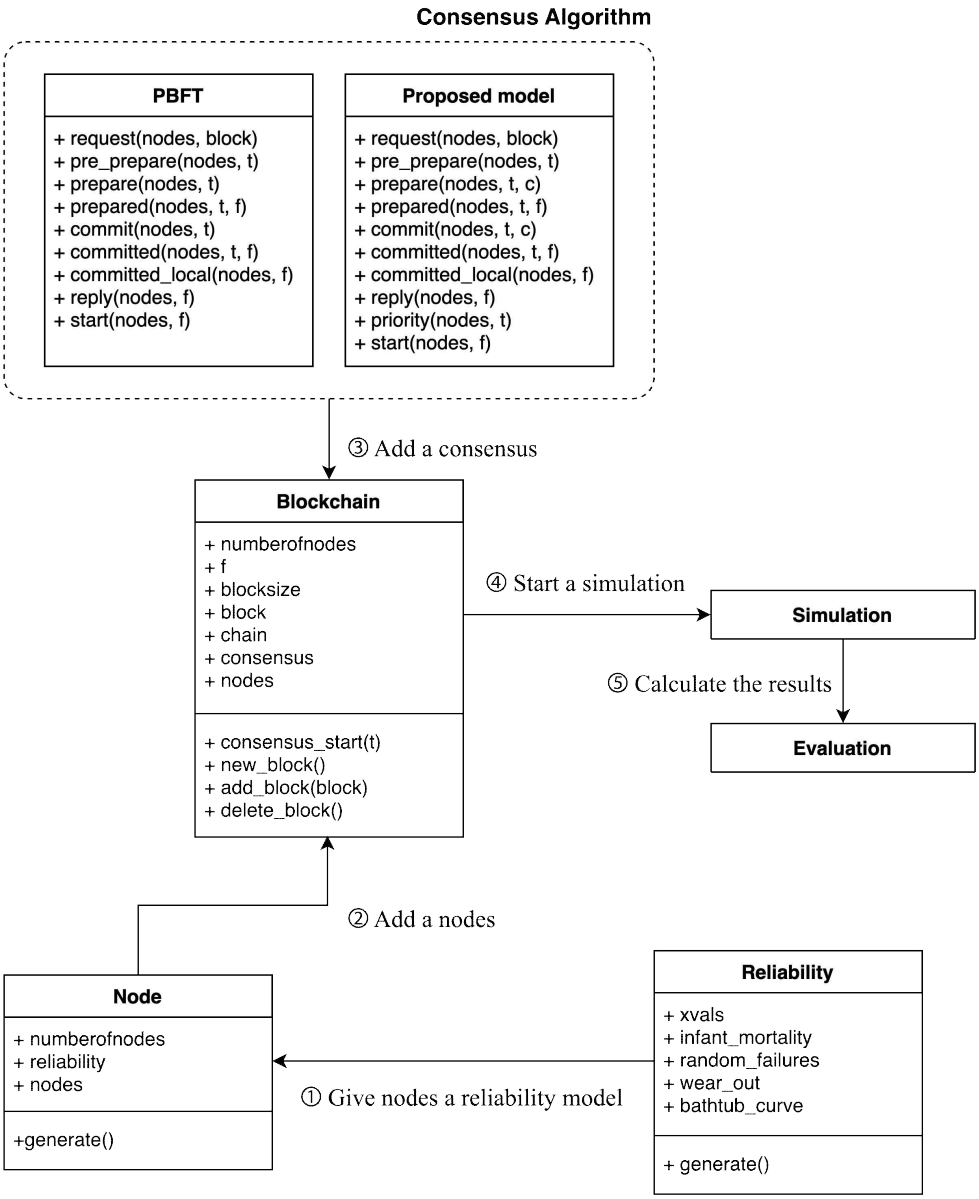

FIGURE 8. Class diagram of evaluation framework

신뢰성 모델 클래스는 Infant mortality, random failures, wear out 모델과 세 모델을 결합한 bathtub curve 모델을 생성할 수 있다. 노드 클래스는 실

험하고자 하는 네트워크의 규모와 신뢰성 모델에 맞추어 노드를 생성한다. 합의 알고리즘 클래스는 PBFT와 제안 방법의 합의 알고리즘을 생성한다. 블록체인 클래스는 앞서 언급한 클래스를 사용해 생성한 실험 요소들을 탑재하여 블록체인의 동작을 수행한다. 시뮬레이션 클래스는 시뮬레이션 조건을 설정하여 반복적인 실험을 수행한다. 평가 클래스는 시뮬레이션 결과값를 받아 지연시간, 처리량, 합의 성공률의 평가 결과를 산출한다.

노드는 클라이언트, 프라이머리 노드, 백업 노드를 고정하여 생성한다. 클라이언트는 블록을 생성하고 합의 시작을 요청할 수 있다. 레플리카는 1부터 n까지의 번호를 부여하여 노드를 구분한다. 1번 노드가 프라이머리 노드, 2번부터 n번까지의 노드가 백업 노드의 역할을 수행한다. 신뢰성 모델은 레플리카 노드의 속성에 추가된다. 노드는 신뢰성 모델을 기반으로 설정된 결함 확률에 따라 정상 동작 또는 결함 동작을 수행한다. 노드가 결함이거나 받은 메시지 개수를 충족하지 못하면 동작을 중단한다.

모든 노드는 서로 연결되어있다고 가정한다. 노드는 모든 노드에게 메시지를 브로드캐스팅하여 합의를 수행한다. 합의 과정에서 발생한 모든 메시지는 노드의 저장소에 기록한다. 저장소에 기록된 메시지를 토대로 메시지 중복, 유효성, 개수를 검증한다. 최종적으로 클라이언트가 reply 메시지를 $f+1$개 전달받았으면 블록체인에 블록을 추가하고, 전달받지 못했다면 블록을 재생성하여 합의를 다시 시도한다.

2. 평가 결과

1) 신뢰성 모델별 평가 결과

먼저 신뢰성 모델별 PBFT와 제안 방법의 평가 결과이다. infant mortality, random failures, wear out 모델에서의 지연시간, 처리량, 합의 성공률을 평가하였다. 실험은 100개의 노드가 1000개의 블록을 생성하는 실험

을 100번 진행한 뒤 평균값을 산출하였다.

그림 9, 10은 신뢰성 모델별 PBFT와 제안 방법의 지연시간, 처리량 및 합의 성공률을 확인할 수 있다.

Infant mortality 모델은 PBFT 합의 알고리즘이 평균 99.6%라는 높은 합의 성공률을 보였다. 하지만 초반의 높은 결함 발생률 때문에 합의 성공률이 급격히 떨어지는 모습을 확인할 수 있었다. 또한 결함으로 인해 지연시간이 증가하고 처리량이 감소한 모습을 보였다. 반면, 제안 모델은 100% 블록 합의에 성공하였다. 지연시간과 처리량은 일정한 수준을 유지하는 것을 확인하였다. 또한, 초반의 높은 결함률을 감내할 수 있어 오히려 합의 규모가 작아지는 효과가 발생하여 지연시간이 낮게 측정되었다.

마찬가지로 wear out 모델은 PBFT가 98.3%의 높은 합의 성공률을 보였지만, 후반부의 높은 결함 발생률 때문에 합의 성공률이 급격히 떨어져 지연시간이 증가하고 처리량이 감소하였다. 반면, 제안 모델은 100% 블록 합의에 성공하였고, 지연시간과 처리량도 일정 수준을 유지할 수 있었다.

Random failures 모델의 낮은 결함 확률은 PBFT도 감내할 수 있는 수준이기 때문에 두 방법 모두 100%의 합의 성공률을 보였다. 지연시간은 제안 방법보다 PBFT가 낮게 측정되었다. 네트워크의 규모가 작고, 결함 발생 확률이 낮은 경우, 제안 방법보다 PBFT를 사용하는 것이 효과적이다.

PBFT는 결함 발생 확률이 낮은 환경, 제안 방법은 결함 발생 확률이 높은 환경에서 효과가 좋음을 알 수 있다. 또한, 시간이 지남에 따라 결함 발생 가능성이 높아지거나 낮아지는 경향성을 보이는 infant mortality과 wear out 모델의 결함 노드를 처리하기 효과적임을 알 수 있다.

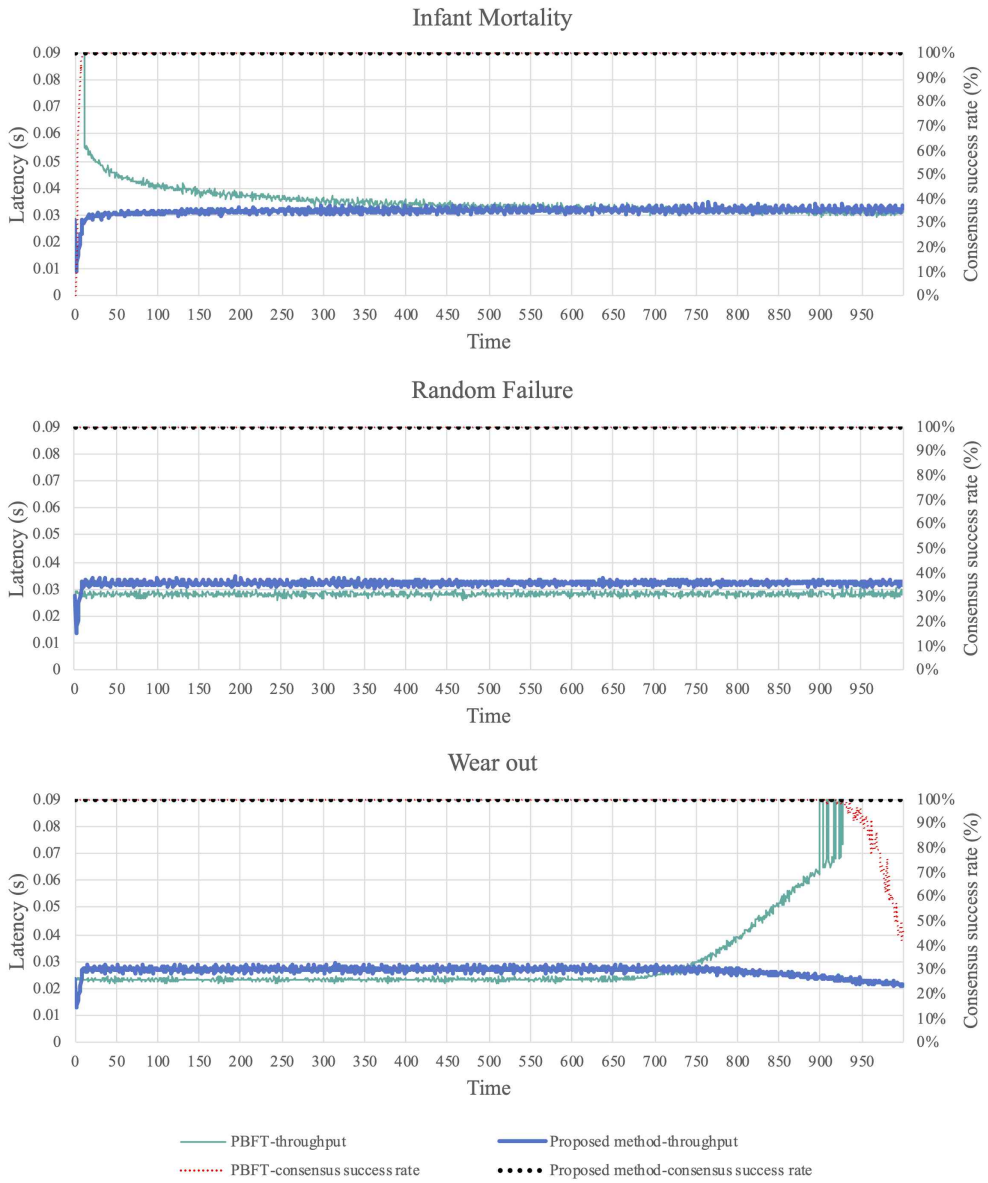

FIGURE 9. Latency and consensus rate of PBFT and proposed method by reliability model

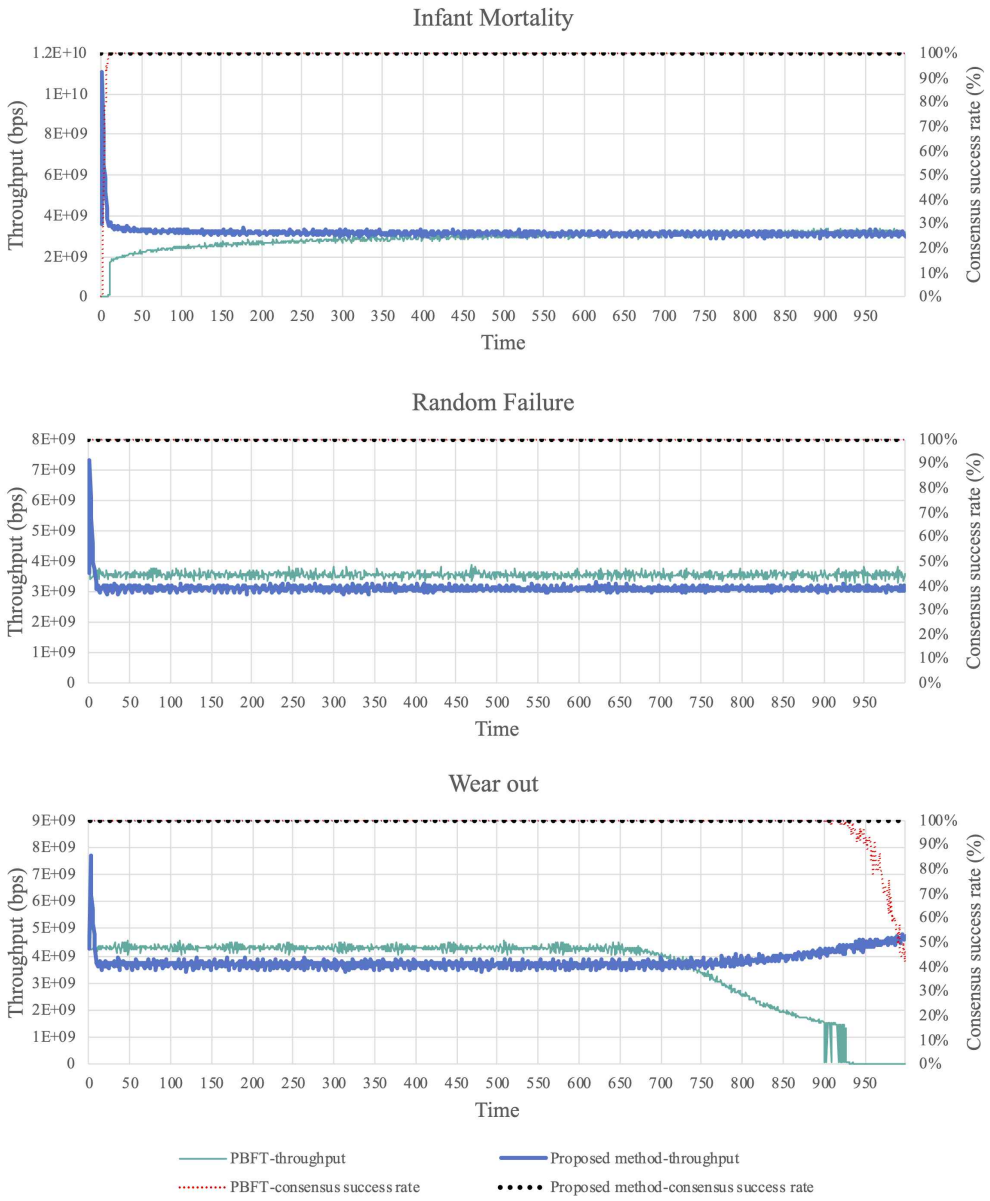

FIGURE 10. Throughput and consensus rate of PBFT and proposed method by reliability model

2) 결함 유무에 따른 평가 결과

다음은 결함 유무에 따른 PBFT와 제안 방법의 평가 결과이다. 결함이 있는 환경과 없는 환경에서의 지연시간, 처리량, 합의 성공률을 평가하였다.

결함이 없는 환경에서는 모든 노드가 결함 없이 정상적으로 동작한다. 결함이 있는 환경에서는 노드의 결함 확률에 따라 결함이 발생한다. 다양한 시기, 다양한 유형의 노드가 모여 있는 현실 환경과 유사한 신뢰성 모델을 부여하기 위해 bathtub curve 신뢰성 모델을 랜덤으로 300단위씩 슬라이싱하여 노드에 부여하였다. 예를 들어 신뢰성 모델이 초기에 치우쳐 있으면 초기 고장을 의미하는 infant mortality, 중기에 치우쳐 있으면 낮고 일정한 확률로 결함을 발생시키는 random failure, 말기에 치우쳐 있으면 기기의 노후화 결함을 반영할 수 있는 wear out 유형의 노드가 생성된다. 실험은 100개의 노드가 300개의 블록을 생성하는 실험을 100번 진행한 뒤 평균값을 산출하였다.

표 7은 결함이 있는 환경에서의 PBFT와 제안 방법의 합의 성공률을 확인할 수 있다. PBFT의 합의 성공률은 96.99%이고 제안 방법의 합의 성공률은 100%이다. PBFT는 결함이 있는 환경에서 100개의 블록을 생성하고자 하였을 때, 약 97개의 블록 합의에 성공할 수 있다.

TABLE Ⅶ

Consensus success rate in fault environments

Consensus algorithm	Consensus success rate (%)
PBFT	96.99%
Proposed method	100%

그림 11, 12는 결함 유무에 따른 PBFT와 제안 방법의 지연시간과 처리량을 확인할 수 있다.

먼저 결함 없는 환경에서 두 방법의 지연시간은 PBFT가 0.018s, 제안 방법은 0.025s로 PBFT 합의 알고리즘을 사용하였을 때 0.007s 빠르다. 반면, 결함 있는 환경에서의 지연시간은 PBFT가 0.05s, 제안 방법이 0.024s로 제안 방법이 0.026s 빠르다. 결함 없는 환경에서 두 방법의 처리량은 PBFT가 2060.56Mbps, 제안 방법은 1484.58Mbps로 PBFT가 575.98Mbps 빠르다. 반면, 결함 있는 환경에서의 처리량은 PBFT가 686.5Mpbs, 제안 방법이 1519.5Mbps로 제안 방법이 833Mbps 빠르다.

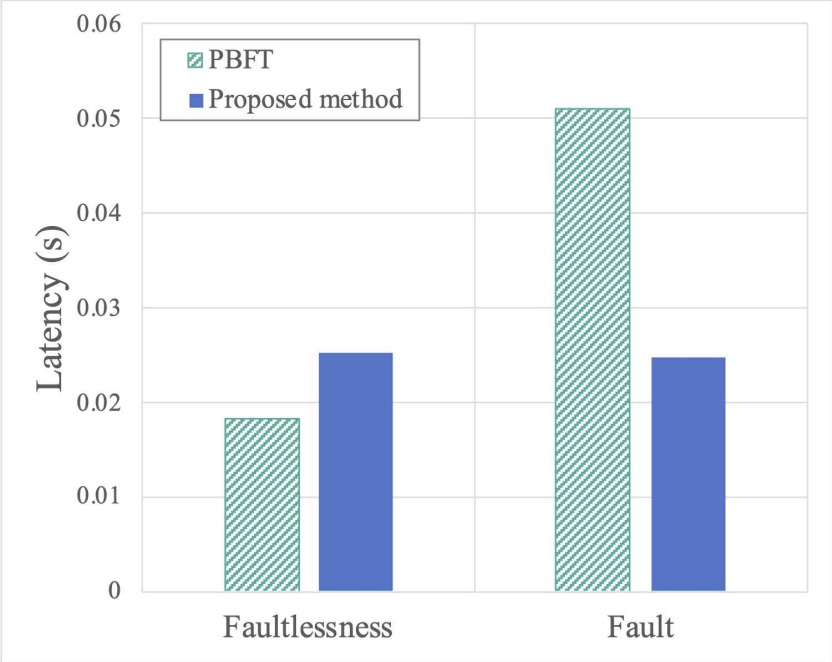

FIGURE 11. Latency of PBFT and proposed method according to fault environments

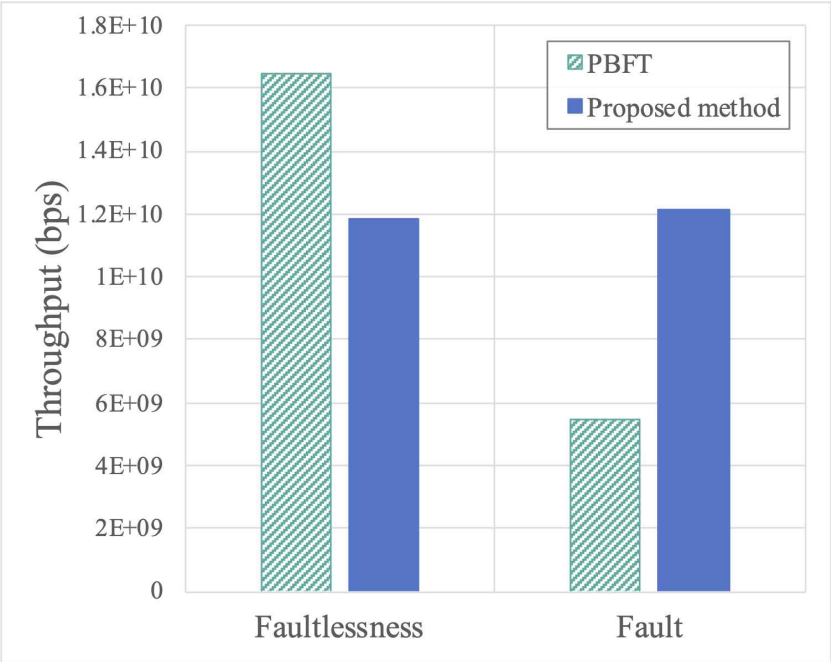

FIGURE 12. Throughput of PBFT and proposed method according to fault environments

결함이 있는 환경과 없는 환경에서 반전된 결과가 나타나는 이유는 두 가지로 추론할 수 있다. 첫 번째로 메시지 수집 대기 시간이다. 노드가 다음 단계로 돌입하기 위해 일정 개수 이상의 메시지를 수집해야 한다. 하지만 결함 노드로 인해 더 오랜 시간 동안 멀리 있는 노드의 메시지를 전달받기 위해 대기해야 한다. 두 번째로 합의 실패 비용이다. 블록 합의에 실패한다면 그 다음 합의의 지연시간으로 축적되기 때문에 지연시간이 증가한다.

Ⅴ. 결론

블록체인은 4차 산업 혁명의 핵심 기술로 금융뿐만 아니라 유통, 문화, 예술, 헬스케어 등 다양한 산업 분야에 응용되고 있다. 분산 네트워크 시스템에서 발생하는 비잔틴 장애를 해결할 수 있는 PBFT는 타 알고리즘 대비 높은 TPS, 절대적 최종성, 적은 에너지 소비량 등 다양한 장점이 있지만 높은 메시지 복잡도로 인한 제한된 확장성으로 대규모 서비스에 적용하기 어려운 한계가 있다. 다양한 방식으로 PBFT의 확장성을 개선하기 위한 연구가 이루어지고 있지만, 탈중앙화, 확장성, 보안성의 트릴레마를 동시에 해결하지 못하고 있다.

본 논문에서는 분산 환경에서 중개 기관 없이 결함 노드를 판단하고 대응하는 방법에 대해 연구하였다. PBFT 합의 과정에서 발생하는 메시지를 통해 결함 정보를 생성하고 인접 노드와 공유하여 노드의 신뢰도를 산출한다. 산출한 신뢰도를 바탕으로 합의 위원회를 조직하여 합의를 진행한다. 제안 방법은 우선순위에 기반한 선별적 합의 참여 기회를 제공하여 확장성을 개선하고, 결함 노드로 인한 지연을 막아 가용성을 향상시켰다. 또한, 분산 환경에서 별도의 대표자를 선출하지 않고 위원회를 구성할 수 있어 탈중앙화의 특성도 보존할 수 있다.

지연시간, 처리량, 합의 성공률의 측면에서 제안 방법을 평가하였다. 제안 방법은 결함이 없는 환경에서는 제안 방법보다 PBFT가 효과적이지만 결함이 있는 환경에서는 제안 방법이 효과적임을 확인하였다. 시간에 따른 인과관계가 있는 초기 발생 결함과 노후화 결함 환경에서 효과적으로 결함 노드를 대응할 수 있음을 확인하였다. 또한, 현실 환경과 유사한 결함 환경에서 지연시간을 0.026s, 처리량을 833Mbps 빠르게 개선하였으며, 합의 성공률을 향상시켰다. 향후 제안 방법의 공간 복잡도와 내결함성에 대해 평가하는 등

더 상세하게 분석할 예정이다. 또한, 공격 모델을 기반으로 제안 방법의 취약성을 평가하고, 보안성을 강화할 수 있는 방법에 대해 연구할 예정이다.

ACKNOWLEDGEMENTS

본 논문은 한국정보통신학회 논문지에 발표한 ‘프랙티컬 비잔틴 장애 허용 기반 블록체인의 확장성과 내결함성 평가 및 비교분석[30]’ 논문의 후속 연구로 수행되었습니다. 논문을 지도해주신 이일구 교수님과 공저자로서 함께 연구에 참여한 김남령, 신나연, 한채림 학생에게 감사드립니다.

참고문헌

[1] Dutta, Pankaj, et al. "Blockchain technology in supply chain operations: Applications, challenges and research opportunities." Transportation research part e: Logistics and transportation review 142 (2020): 102067.

[2] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." Decentralized Business Review (2008): 21260.

[3] Lamport, Leslie, Robert Shostak, and Marshall Pease. "The Byzantine generals problem." ACM Transactions on Programming Languages and Systems 4.3 (1982): 382-401.

[4] Castro, Miguel, and Barbara Liskov. "Practical byzantine fault tolerance." OsDI. Vol. 99. No. 1999. 1999.

[5] Gao, Sheng, et al. "T-PBFT: An EigenTrust-based practical Byzantine fault tolerance consensus algorithm." China Communications 16.12 (2019): 111-123.

[6] The Scalability Trilemma in Blockchain, Sep. 2019, [online] Available: https://medium.com/@aakash_13214/the-scalability-trilemma-in-blockchain-75fb57f646df.https://medium.com/@aakash_13214/the-scalability-trilemma-in-blockchain-75fb57f646df.

[7] Fischer, Michael J., Nancy A. Lynch, and Michael S. Paterson. "Impossibility of distributed consensus with one faulty process." Journal of the ACM (JACM) 32.2 (1985): 374-382.

[8] Mingxiao, Du, et al. "A review on consensus algorithm of

blockchain." 2017 IEEE international conference on systems, man, and cybernetics (SMC). IEEE, 2017.

[9] Xiao, Yang, et al. "A survey of distributed consensus protocols for blockchain networks." IEEE Communications Surveys & Tutorials 22.2 (2020): 1432-1465.

[10] Zhang, Shijie, and Jong-Hyouk Lee. "Analysis of the main consensus protocols of blockchain." ICT express 6.2 (2020): 93-97.

[11] Li, Peilun, et al. "Gosig: a scalable and high-performance byzantine consensus for consortium blockchains." Proceedings of the 11th ACM Symposium on Cloud Computing. 2020.

[12] Yang, Yin. "Linbft: Linear-communication byzantine fault tolerance for public blockchains." arXiv preprint arXiv:1807.01829 (2018).

[13] Li, Zhong-Cheng, et al. "ISCP: An Improved Blockchain Consensus Protocol." Int. J. Netw. Secur. 21.3 (2019): 359-367.

[14] Jalalzai, Mohammad M., Costas Busch, and Golden G. Richard. "Proteus: a scalable BFT consensus protocol for blockchains." 2019 IEEE international conference on Blockchain (Blockchain). IEEE, 2019.

[15] Jalalzai, Mohammad M., Costas Busch, and Golden G. Richard. "Consistent BFT performance for blockchains." 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S). IEEE, 2019.

[16] Rashid, Mohammed A., and Houshyar Honar Pajooh. "A security framework for IoT authentication and authorization based on blockchain technology." 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th

IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). IEEE, 2019.

[17] Biswas, Sujit, et al. "PoBT: A lightweight consensus algorithm for scalable IoT business blockchain." IEEE Internet of Things Journal 7.3 (2019): 2343-2355.

[18] Li, Wenyu, et al. "A scalable multi-layer PBFT consensus for blockchain." IEEE Transactions on Parallel and Distributed Systems 32.5 (2020): 1146-1160.

[19] Tong, Wei, Xuewen Dong, and Jiawei Zheng. "Trust-pbft: A peertrust-based practical byzantine consensus algorithm." 2019 International Conference on Networking and Network Applications (NaNA). IEEE, 2019.

[20] Fortino, Giancarlo, et al. "Using blockchain in a reputation-based model for grouping agents in the Internet of Things." IEEE Transactions on Engineering Management 67.4 (2019): 1231-1243.

[21] She, Wei, et al. "Blockchain trust model for malicious node detection in wireless sensor networks." IEEE Access 7 (2019): 38947-38956.

[22] Xu, Guangquan, et al. "SG-PBFT: A secure and highly efficient distributed blockchain PBFT consensus algorithm for intelligent Internet of vehicles." Journal of Parallel and Distributed Computing 164 (2022): 1-11.

[23] Kim, Tai-Hoon, et al. "A novel trust evaluation process for secure localization using a decentralized blockchain in wireless sensor networks." IEEE access 7 (2019): 184133-184144.

[24] Chen, Jing, et al. "Certchain: Public and efficient certificate audit based on blockchain for tls connections." IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2018.

[25] Heo, Hoon-Sik, and Dae-Young Seo. "A study on scalable PBFT consensus algorithm based on blockchain cluster." The Journal of The Institute of Internet, Broadcasting and Communication 20.2 (2020): 45-53.

[26] Li, Suisheng, et al. "Blockchain dividing based on node community clustering in intelligent manufacturing cps." 2019 IEEE International Conference on Blockchain (Blockchain). IEEE, 2019.

[27] Khakurel, Utsab, Danda Rawat, and Laurent Njilla. "FastChain: Lightweight blockchain with sharding for Internet of battlefield-things in NS-3." 2019 IEEE International Conference on Industrial Internet (ICII). IEEE, 2019.

[28] Choi, Beongjun, et al. "Scalable network-coded PBFT consensus algorithm." 2019 IEEE International Symposium on Information Theory (ISIT). IEEE, 2019.

[29] Moubray, John. Reliability-centered maintenance. Industrial Press Inc., 2001.

[30] Lee, Eun-Young, et al. "Evaluation and Comparative Analysis of Scalability and Fault Tolerance for Practical Byzantine Fault Tolerant based Blockchain." Journal of the Korea Institute of Information and Communication Engineering 26.2 (2022): 271-277.

ABSTRACT

Determining and responding to faulty nodes in a distributed environment to improve PBFT scalability

Eun-Young Lee
Department of Future Convergence
Technology Engineering
Graduate School of
Sungshin University

PBFT (Practical Byzantine Fault Tolerant) is a consensus algorithm that can achieve the consensus of the entire network by resolving unintentional and intentional faults in a distributed network environment, guaranteeing high performance and absolute finality. However, the load increases exponentially as the size of the network increases due to the amount of messages transmitted during the consensus process. In addition, previous studies aimed at improving PBFT have limitations in that they cannot simultaneously solve the trilemma of blockchain: decentralization, scalability, and security.

In this paper, a method for determining and responding to faulty nodes in a distributed environment for improving the performance of PBFT is proposed. The proposed method collects messages generated during the PBFT consensus process to determine the reliability of the node, and organizes a small committee according to the reliability level to proceed

with the consensus. Scalability can be improved by providing an opportunity to participate in selective consensus based on priority, and availability can be improved by preventing delays caused by faulty nodes. In addition, the characteristics of decentralization can be preserved by forming a committee without electing a separate representative in a distributed environment. In this paper, latency, throughput, and consensus success rate of PBFT and the proposed method were evaluated. According to the evaluation results, the proposed method effectively responded to defects in a defect environment with a causal relationship over time. In addition, although PBFT is more effective than the proposed method in a faultlessness environment, it was confirmed that the delay time was improved by 0.026 s and the throughput by 833 Mbps, and the consensus success rate was also improved in a fault environment.