BAFFLE

BLOCKCHAIN BASED
AGGREGATOR FREE FEDERATED LEARNING

REFERENCE

- Ramanan, Paritosh, and Kiyoshi Nakayama.
  "Baffle: Blockchain based aggregator free federated learning."
  2020 IEEE International Conference on Blockchain (Blockchain). IEEE, 2020.

ABSTRACT

ABSTRACT

- 연합 학습 (Federated Learning, FL)
  - 글로벌 모델을 유지하고 업데이트하는
  - 중앙화된 집계자(aggregator)가 존재해야 함
- 많은 경우에 이런 집계자의 존재가 비실현적임
  - 연산이 많이 들기 때문

ABSTRACT

- BAFFLE
  - 집계자 없는 블록체인 기반 FL 환경
  - 본질적으로 탈중앙화
- 스마트 컨트랙트를 사용
  - 라운드 관리, 모델 통합, FL 업무(task) 업데이트

ABSTRACT

- 연산 성능의 향상
  - 점수(score) 및 제안(bid) 전략에 따라
  - 글로벌 파라미터를 구분된 청크(chunk)들로 분해한 덕

ABSTRACT

- 프라이빗 이더리움 네트워크 상에서 실험 진행
- 중앙화된 집계자가 존재하는 상황과 비교
- BAFFLE이 현저하게
  - FL을 위한 가스 비용을 줄임
- 중앙화 상황과 유사한 정확도를 가지면서도
  - 높은 확장성과 연산 효율성을 가짐

INTRODUCTION

INTRODUCTION

- FL
  - FL 패러다임의 기본 가정은 중앙화된 집계자가 존재한다는 것
- 집계자는 네 가지 핵심 기능을 수행함
  - 1) 각 라운드마다 글로벌 연산(통합)을 수행해야 함
  - 2) 글로벌 머신 러닝 모델을 유지
  - 3) 글로벌 모델을 선택(선별)된 엔드유저에게 전송
  - 4) 글로벌 모델을 선택(선별)된 유저들의 모델을 이용해 업데이트

INTRODUCTION

- 중앙화된 집계자의 존재는 문제가 될 수 있음
  - 비실현적일 수 있음
  - 엔드유저들이 무조건적으로 신뢰해야 함
  - 강인하지 않음: Single point of failure
  - 클라우드 기반인 경우가 많음
    - 확장성이 떨어지거나
    - 비쌈

INTRODUCTION

- 블록체인 기반 탈중앙화는 집계자의 중앙화 문제를 잘 다룰 수 있음
- 그러나 고려해야 할 점들이 있음
  - 블록체인에서의 데이터 스토리지 및 연산 비용이 높음
  - 머신 러닝 모델 전체를 블록체인에서 다루기에는 부담이 큼: 레이턴시 등
  - 트랜잭션 크기의 제한으로부터 다룰 수 있는 데이터 크기 제한이 있음
- FL에서 모델을 저장하거나 업데이트하는 것에 제약사항이 됨

INTRODUCTION: BAFFLE

- BAFFLE
  - 블록체인 기반 집계자 없는 FL 환경
  - 스마트 컨트랙트(Smart Contract, SC)로 글로벌 모델을 유지하고
  - 유저의 상태를 관리

INTRODUCTION: BAFFLE

- BAFFLE에서 사용자들이
  - 독립적이고 병렬적으로
  - 낮은 연산 비용으로
  - SC에 있는 글로벌 모델을 업데이트할 수 있도록 함
- 특정 라운드에서 엔드 유저를 선별하는 방법
  - SC에서 평가한
  - 엔드 유저 로컬 업데이트의 가치에 따라 선별됨

RELATED WORK

RELATED WORK

- R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pp. 1310-1321, ACM, 2015.
  - 프라이버시를 보존하며 분산 데이터를 사용하는
  - 글로벌 신경망 모델을 증진시키는 방법에 대한 연구
  - 글로벌 모델의 파라미터가 저장되는 글로벌 공유 메모리 모델을 고려
  - 참여자는 로컬 훈련을 기반으로, 글로벌 파라미터의 랜덤 서브셋을 업데이트할 수 있음

RELATED WORK

- FL에 대한 연구 및 이론적 설명에 관한 연구들
  - 본 논문의 references [1], [12], [13]
- 최근에는 여러 집계자가 존재하는 구조도 존재[2]
  - 분산 집계자 관점을 제시했으나
  - 집계자 free 환경은 아님

RELATED WORK

- 완전히 탈중앙화된 FL
  - A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar,
    “Fully decentralized federated learning,”
    Proceedings of third workshop on Bayesian Deep Learning (NeurIPS), 2018.
  - A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar,
    “Peer-to-peer federated learning on graphs,”
    arXiv preprint arXiv:1901.11173, 2019.
  - 사용자가 믿음(belief)을 이웃들로부터 수집한 정보로 업데이트

RELATED WORK

- 블록체인과 AI의 결합
- Cortex
  - SC 기반 머신 러닝 플랫폼
- DeepBrain Chain
  - 채굴 노드가 AI 모델을 증진함으로써 이득을 얻음

RELATED WORK

- 블록체인과 FL의 결합
- H. Kim, J. Park, M. Bennis, and S.-L. Kim,
  “On-device federated learning via blockchain and its latency analysis,”
  - FL 프레임워크를 채굴 메커니즘에 통합
  - 기본 합의 프로토콜을 수정해야 하므로 구현하기가 어려움
- B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas,
  “Communication-efficient learning of deep networks from decentralized data,”
  - 블록체인에 훈련 데이터를 공시해야하는 한계가 있음

RELATED WORK

- BAFFLE
  - 이더리움과 같은 현존하는 블록체인과 호환되는
  - production-level 탈중앙화 FL 플랫폼
- To the best of our (저자들의) knowledge, 첫 번째 시도

SMART CONTRACT

SMART FL CONTRACT DESIGN: DECENTRALIZING ROLE OF AGGREGATOR

- FL 과정을 집계자 free하게 만들기 위해서는
  - 기술적 요소들이 필요함
- BAFFLE이 이더리움 플랫폼 위에서 구현되고 평가되었지만
  - 다른 블록체인 기반 SC에서도 잘 구동되도록 확장 가능할 것

CHUNKING

- 대부분의 블록체인 플랫폼이
  - 각 트랜잭션의 데이터 사이즈에
  - 상한을 가지고 있음
- 논문에서 사용한 버전의 이더리움 가상 머신 (Ethereum Virtual Machine, EVM) 기준
  - 24 kB의 제한이 있음
  - 이는 집계자 free FL 스킴의 병목이 됨

CHUNKING

- 머신 러닝 모델은 통상 블록 크기보다 큼
  - 머신 러닝 모델 가중치 벡터를 여러 청크(chunk)로 나눌 필요가 있음
  - 각 청크 사이즈는 최대 트랜잭션 사이즈보다 작아야 함
- 그러나 청킹(chunking)은 몇 가지 고려사항을 요구함
  - 1. 직렬화
  - 2. 예산

CHUNKING

- 1. 직렬화 (압축)
  - SC의 스토리지가 비싸기 때문에
  - 머신 러닝은 직렬화된 포맷으로 저장될 필요가 있음
  - 그러나 직렬화 이후 모델을 파티셔닝하는건 모순적
- 따라서 어떠한 FL 업무에 대해
  - 우선 파티셔닝을 하고
  - 개별적으로 직렬화해야 함

CHUNKING

- 1. 직렬화 (압축)
- 이러한 chunk-and-serialize 스킴은 여러 강점을 가짐
  - 청크는 개별적으로 읽거나 쓰일 수 있음
  - 병렬적 업데이트를 가능하게 함
  - 모델의 일부가 가지는 가치를 평가할 수 있음

CHUNKING

- 2. 예산
  - 청킹의 잠재적 이익
    - 사용자 디바이스가 개별적으로 기여 수준을 결정짓도록 함
  - 블록체인에 청크를 입력하는 것은 수수료 등의 연산 비용을 수반하므로
    - 사용자들은 개별적으로 이익/비용 비율을 평가하고
    - 업데이트하고자 하는 청크의 개수를 정할 수 있음

CHUNKING

- 2. 예산
  - 사용자 디바이스별로 업데이트하고자 하는 청크의 최대 수는 차이가 남
    - 예산에 따라
  - 본질적으로 서로 다른 예산을 가지는 상황을 상정하게 될 것

BIDDING AND ROUND DELINEATION

- 각 청크는 점수가 할당됨
  - 엔드 유저 디바이스로부터
  - 최신 글로벌 카피와의 노름(norm) 차이에 기반해서

BIDDING AND ROUND DELINEATION

- 랜덤 선택에 따라
  - 사용자 디바이스는 청크 집합에 대한 제안(bid)을 제출
    - 사용자의 예산이 허가하는 한도 내에서
- SC는
  - 모든 라운드에서
  - 서로 다른 사용자 디바이스로부터
  - 다양한 제안들을 받음

BIDDING AND ROUND DELINEATION

- 여러 제안이 등록된 청크에 대해
  - 최대 점수를 등록한 디바이스가 그 청크의 업데이터로 선별됨

BIDDING AND ROUND DELINEATION

- 참여 수준 (Participant Level, PL)
  - 모든 FL 학습 업무에서 선택됨
  - (다음) 라운드 시작을 위해 제안을 등록해야 할 에이전트의 수
- PL이 만족되면
  - (다음) 라운드가 시작되고
  - (이전 라운드에) 새 디바이스는 참여할 수 없음

COMPUTATION

COMPUTATIONAL PERSPECTIVES OF BAFFLE

- BAFFLE은 두 중요한 관점을 가짐
  - 로컬 관점: 사용자 디바이스가 취하는 연산 스텝, 이들과 블록체인 간 상호작용
  - 글로벌 관점: SC 기반, 다양한 FL 단계를 수행

USER ORIENTED LOCAL COMPUTATION

- 클라이언트 프로세스는 로컬 학습을 수행한 후
  - SC를 활용하여 로컬 모델을 블록체인으로 입력

USER ORIENTED LOCAL COMPUTATION

- 로컬 학습
  - 사용자 디바이스가 SC를 통해 최신 접근 가능한 모델을 가져옴
  - 이 모델은 최신 로컬 카피들의 평균
  - 이 모델을 로컬 데이터를 가지고 훈련, 새 로컬 모델을 얻음
  - 새 모델은 다음 라운드에서 블록체인에 입력될 후보가 됨

USER ORIENTED LOCAL COMPUTATION

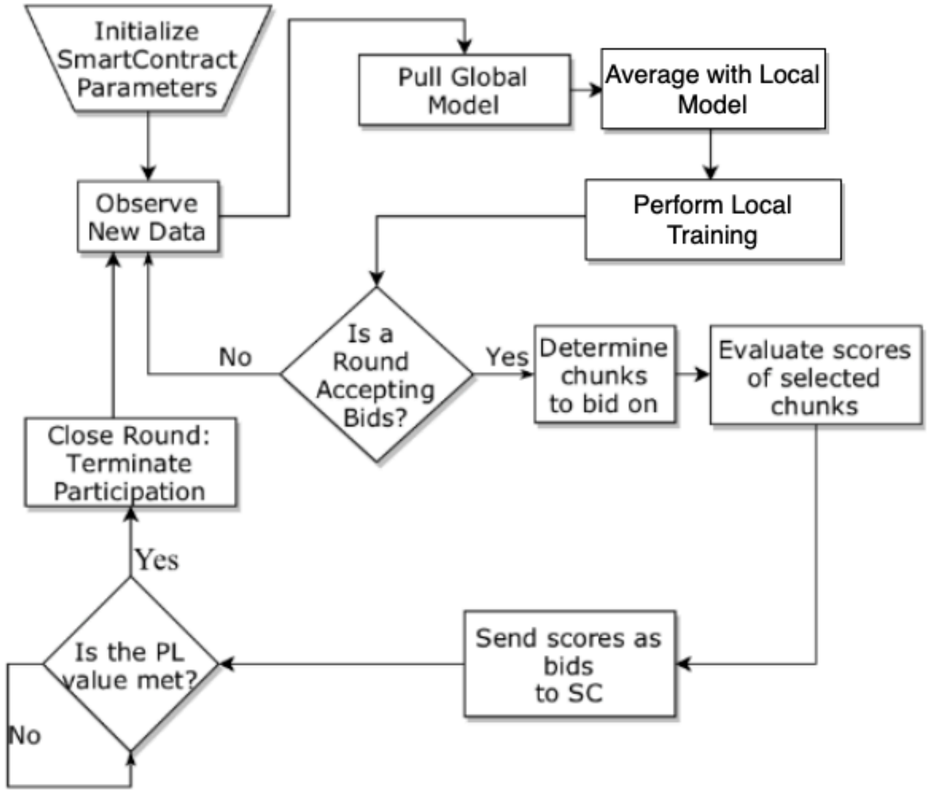
- 로컬 모델을
  - 다른 디바이스들(의 로컬 모델)과 통합하고
  - 이 업데이트를 체인에 입력해야 함

USER ORIENTED LOCAL COMPUTATION

- 수도코드와 흐름도

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^{j,c}}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subseteq \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in C^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for

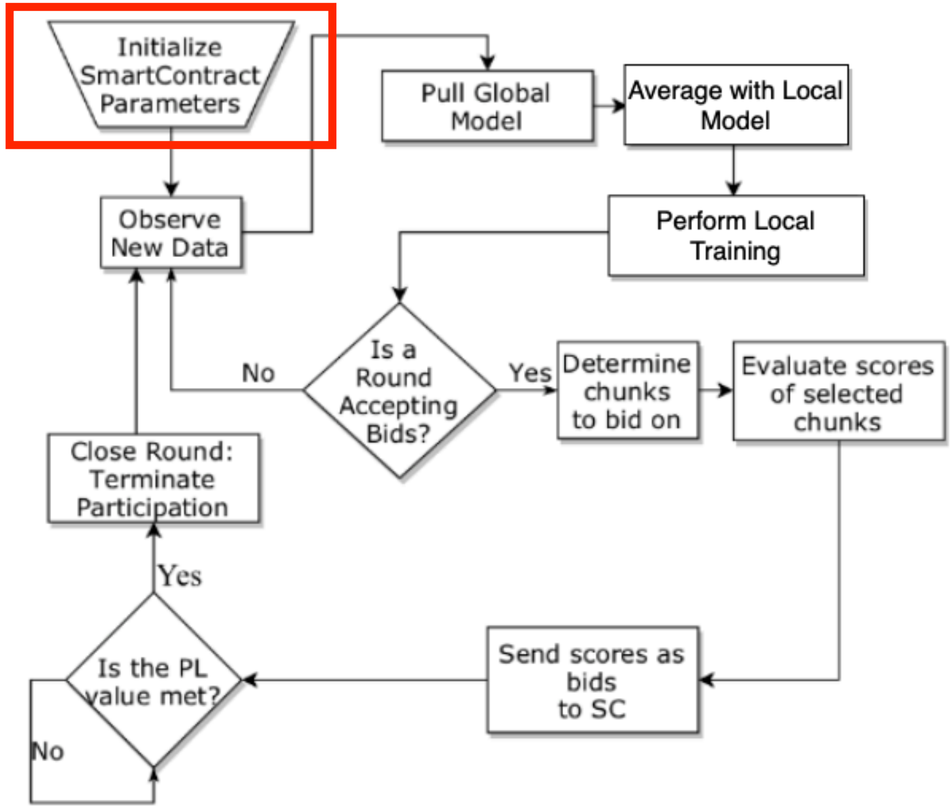


USER ORIENTED LOCAL COMPUTATION

- 각 디바이스는 청크들의 집합 C와 최대 예산 B로 초기화됨

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^{j,c}}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subseteq \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in C^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for

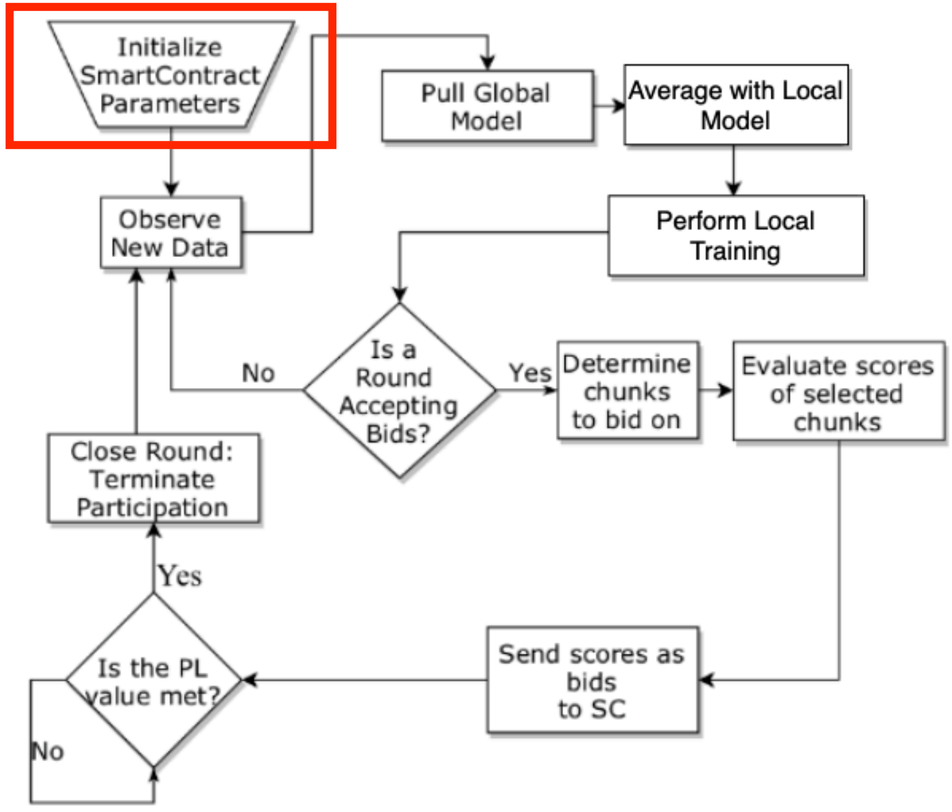


USER ORIENTED LOCAL COMPUTATION

- Q^c : SC에 있는 청크 c에 포함된 파라미터

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^{j,c}}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subseteq \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in C^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for

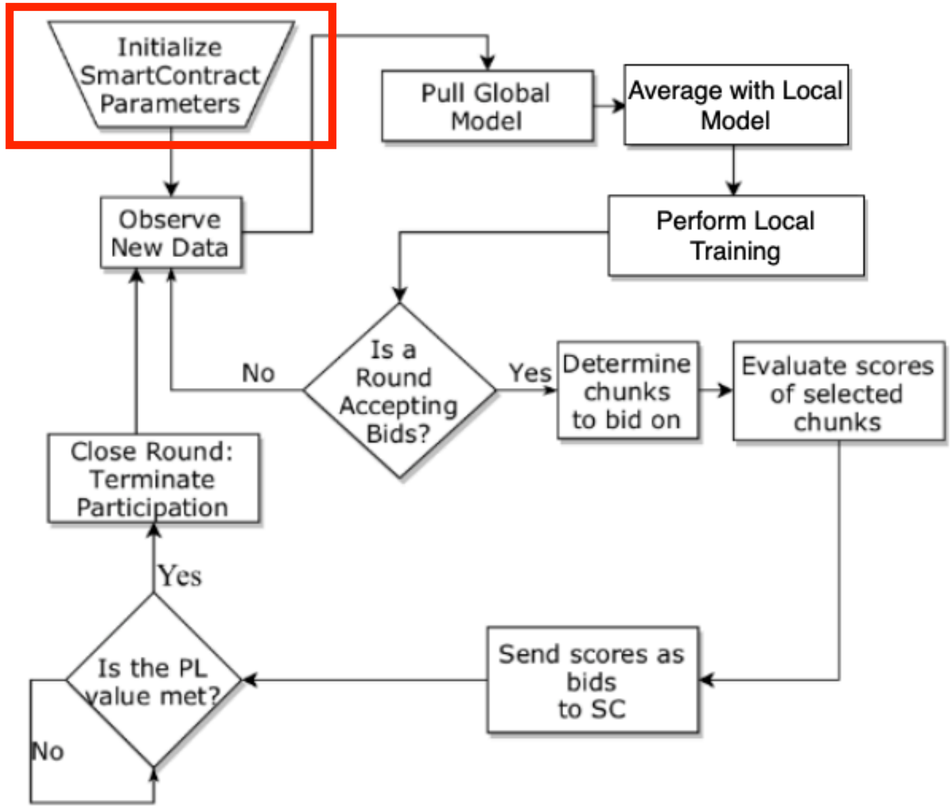


USER ORIENTED LOCAL COMPUTATION

- $Q^{j,c}$: 청크 c에 대한 에이전트 j의 파라미터

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^{j,c}}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subseteq \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in C^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for

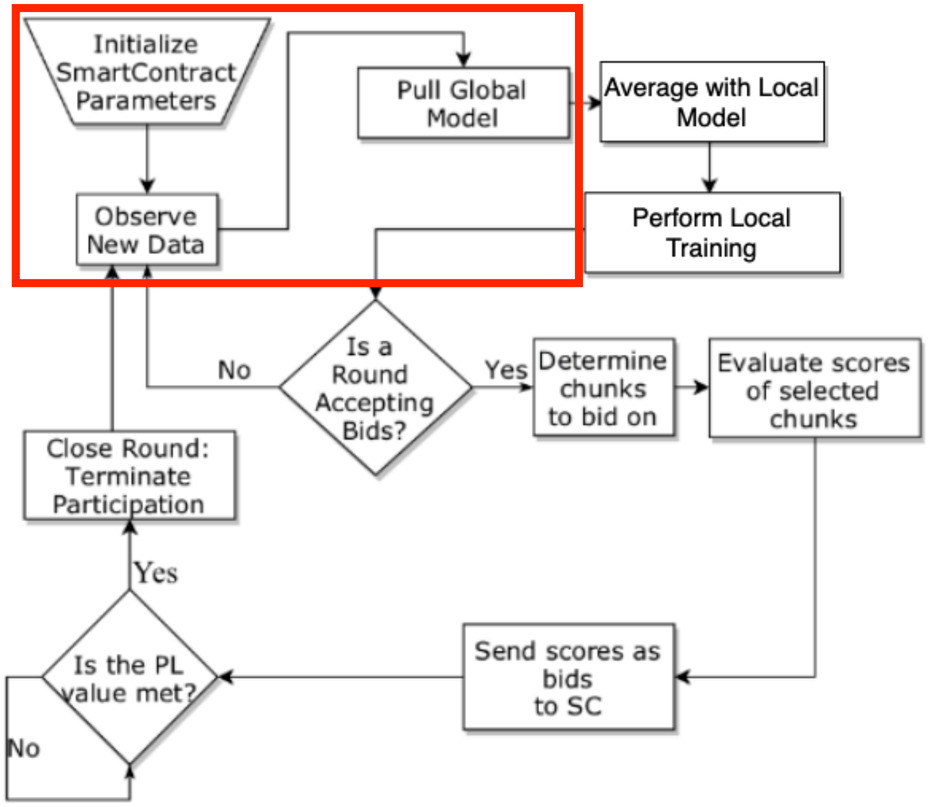


USER ORIENTED LOCAL COMPUTATION

- 라운드 k에 대해, 사용자 디바이스는 글로벌 모델 카피를 가져옴

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^j}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subseteq \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in C^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for

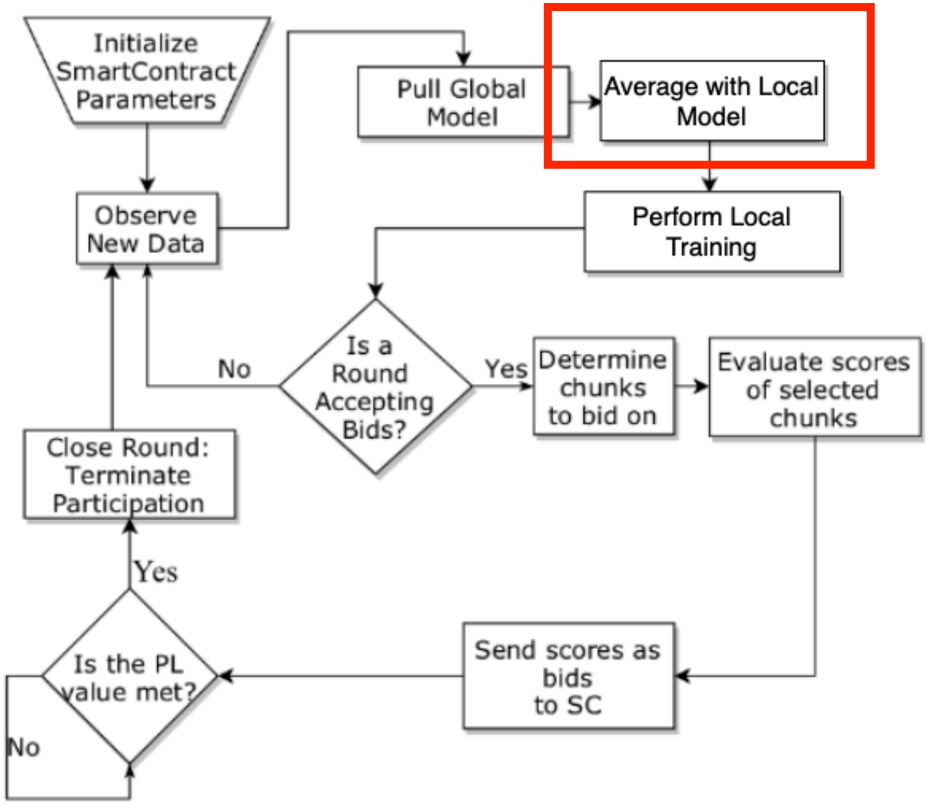


USER ORIENTED LOCAL COMPUTATION

- 로컬 카피와 평균냄

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^{j,c}}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subseteq \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in C^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for

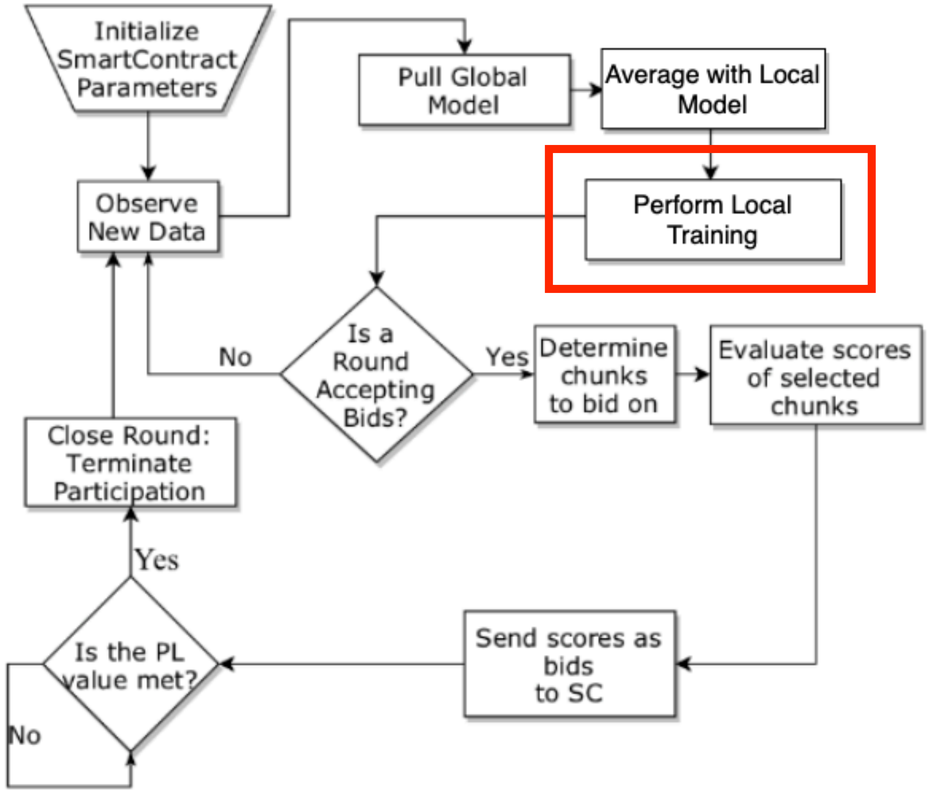


USER ORIENTED LOCAL COMPUTATION

- 로컬 훈련을 수행하고 Q^j_{k+1}을 업데이트

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q^j_0
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q^{j,c}_{k+1} \leftarrow \frac{Q^c + Q^{j,c}_k}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q^j_{k+1}
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subseteq \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q^{j,c}_{k+1}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in C^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for

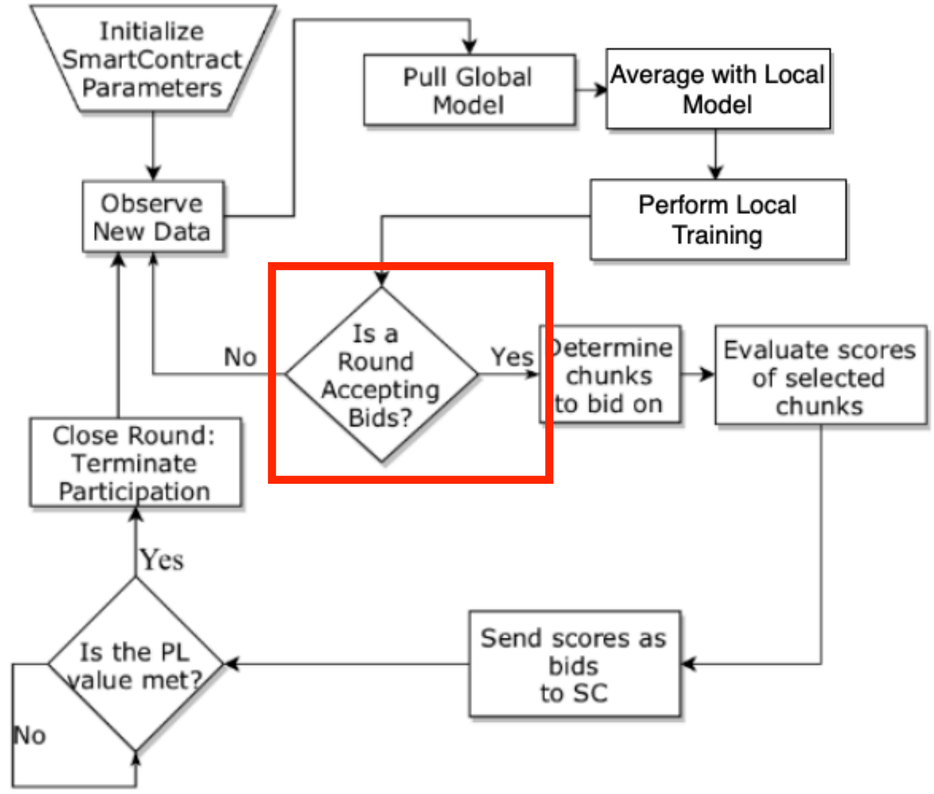


USER ORIENTED LOCAL COMPUTATION

- 라운드가 가용한 상태라면(제안을 받는다면)

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^{j,c}}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $C^k \subseteq \mathcal{C}, |C^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in C^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for

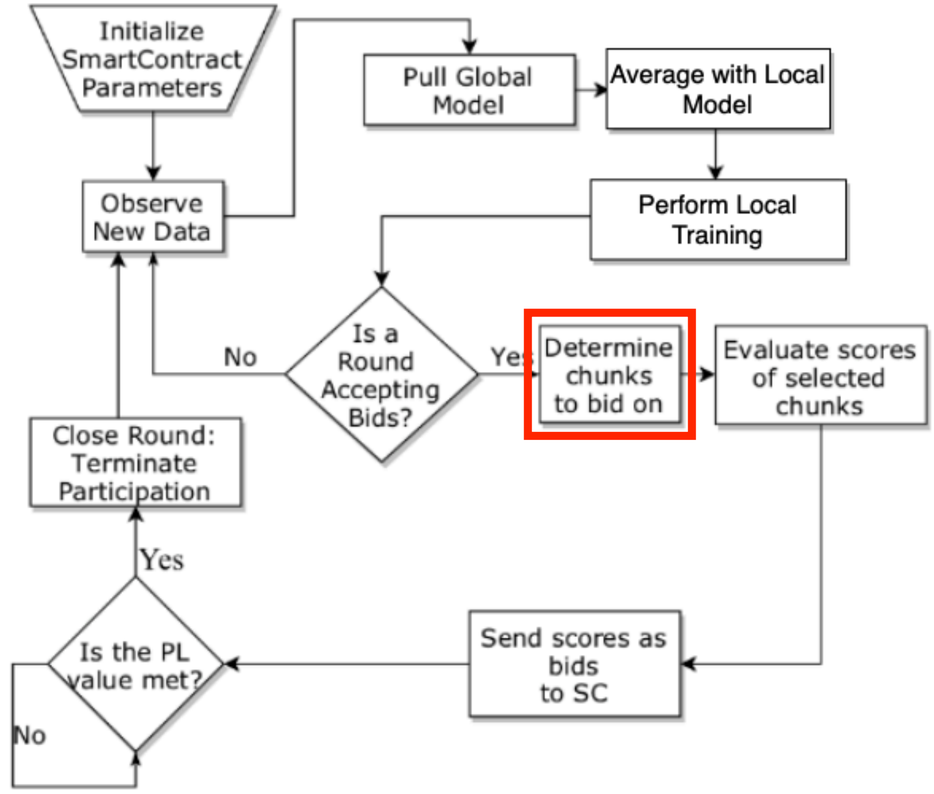


USER ORIENTED LOCAL COMPUTATION

- 예산 크기에 맞춰 로컬 청크에서 무작위로 선별

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^{j,c}}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subseteq \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in C^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for

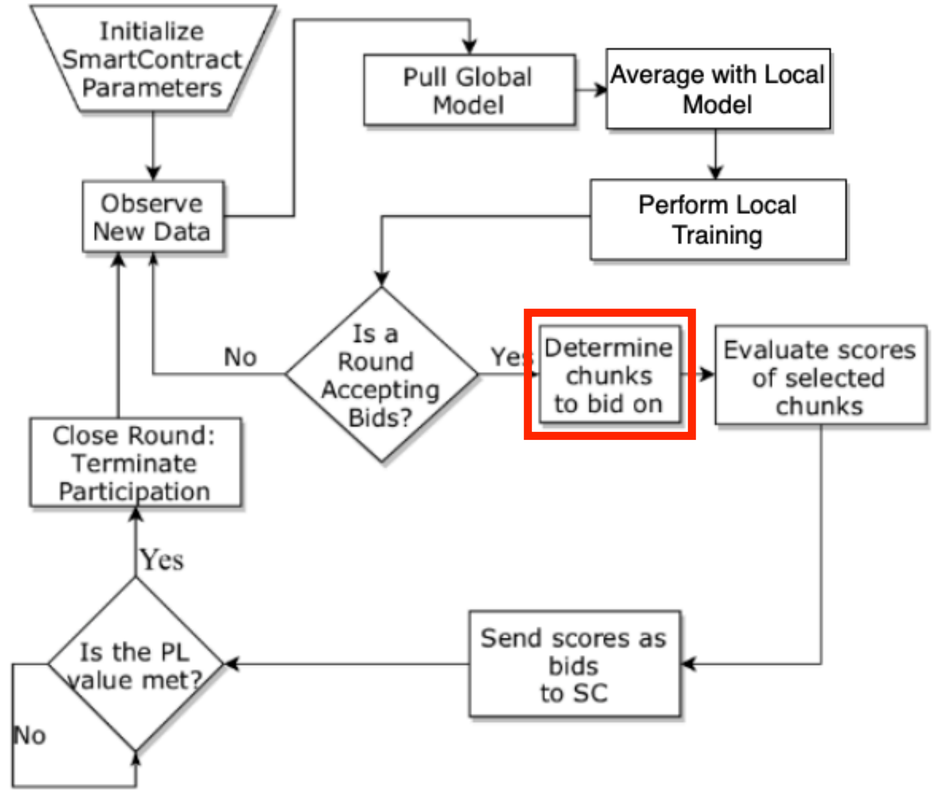


USER ORIENTED LOCAL COMPUTATION

- B개 중 청크 c를 포함할 확률은 B/C

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^{j,c}}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subseteq \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in C^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for

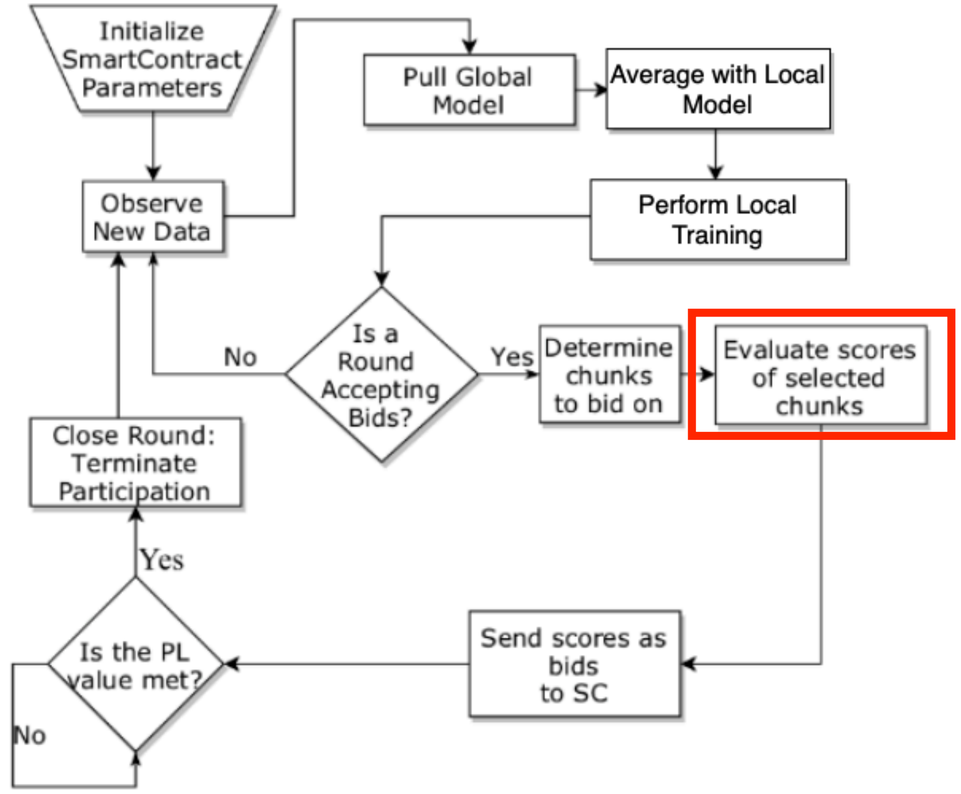


USER ORIENTED LOCAL COMPUTATION

- 점수 δ는 각 청크에 할당됨: 글로벌 모델과 로컬 (업데이트된) 모델의 거리

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^{j,c}}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subset \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in \tilde{C}^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for

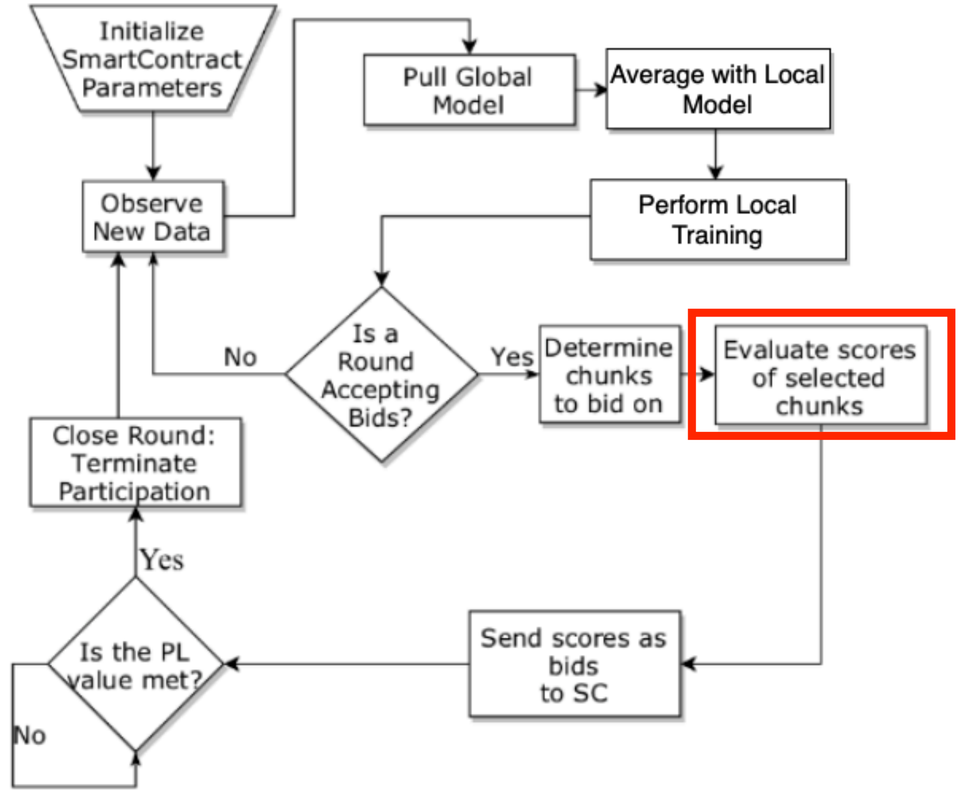


USER ORIENTED LOCAL COMPUTATION

- 이 점수는 SC에 등록될 제안의 기반이 됨

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^{j,c}}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subset \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in \tilde{C}^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for

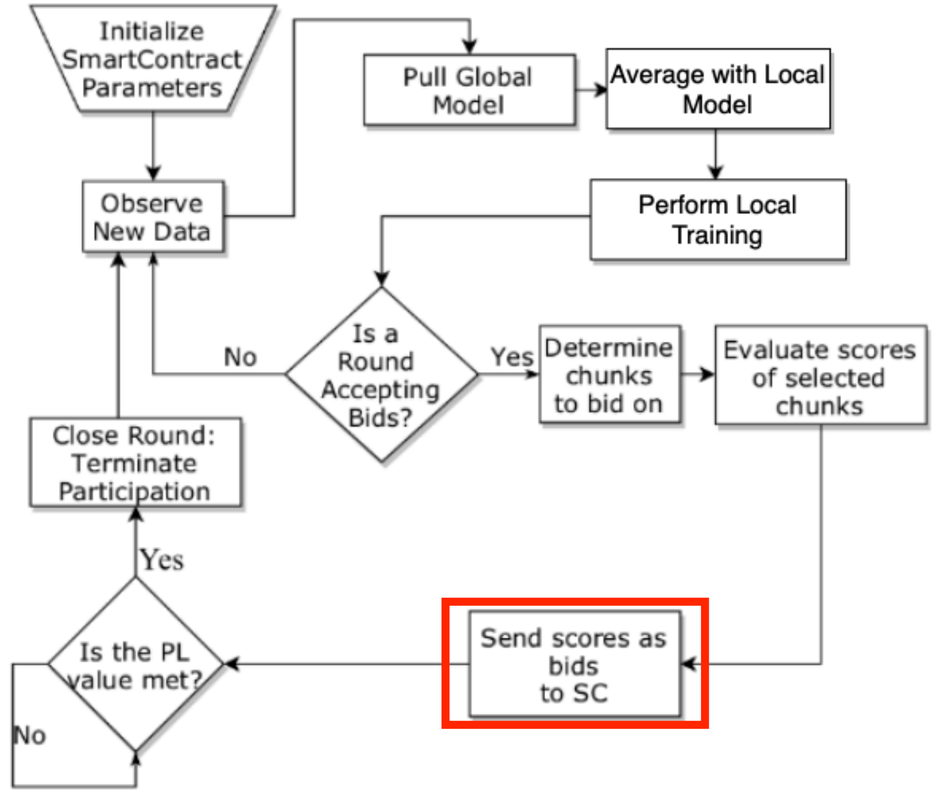


USER ORIENTED LOCAL COMPUTATION

- 여러 디바이스가 같은 청크에 제안을 할 경우 (Luke's comment: '같은 점수로' 제약 포함) 누가 승리(winning)할 것인지는 uniform하다 가정

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^{j,c}}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subseteq \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in C^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for



USER ORIENTED LOCAL COMPUTATION

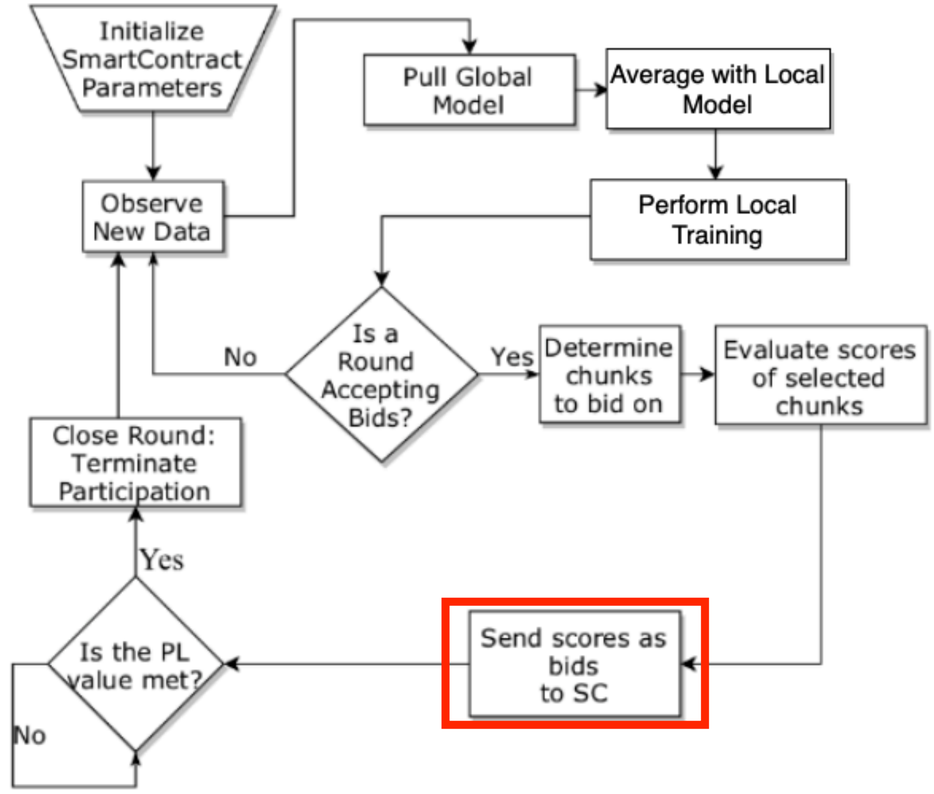
- 무작위로 선별한 B개 중 청크 c를 포함할 확률은 B/C 이고
  - 승리(winning)할 확률은 uniform하다 가정했으므로
  - 이항 분포를 사용하면
  - 특정 라운드에 디바이스 j가 청크 c를 업데이트할 확률은
    - $\mu \cdot \frac{B}{C}$, where $\mu = \sum_{d=0}^{L-1} \frac{1}{d+1}\binom{L-1}{d}\left(\frac{B}{C}\right)^{d} \cdot \left(1-\frac{B}{C}\right)^{L-d-1}$
    - L은 라운드에 참여한 참여자의 수 (PL 값)

USER ORIENTED LOCAL COMPUTATION

- 블록체인에 선택한 C^k를 등록

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^{j,c}}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subseteq \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in C^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for

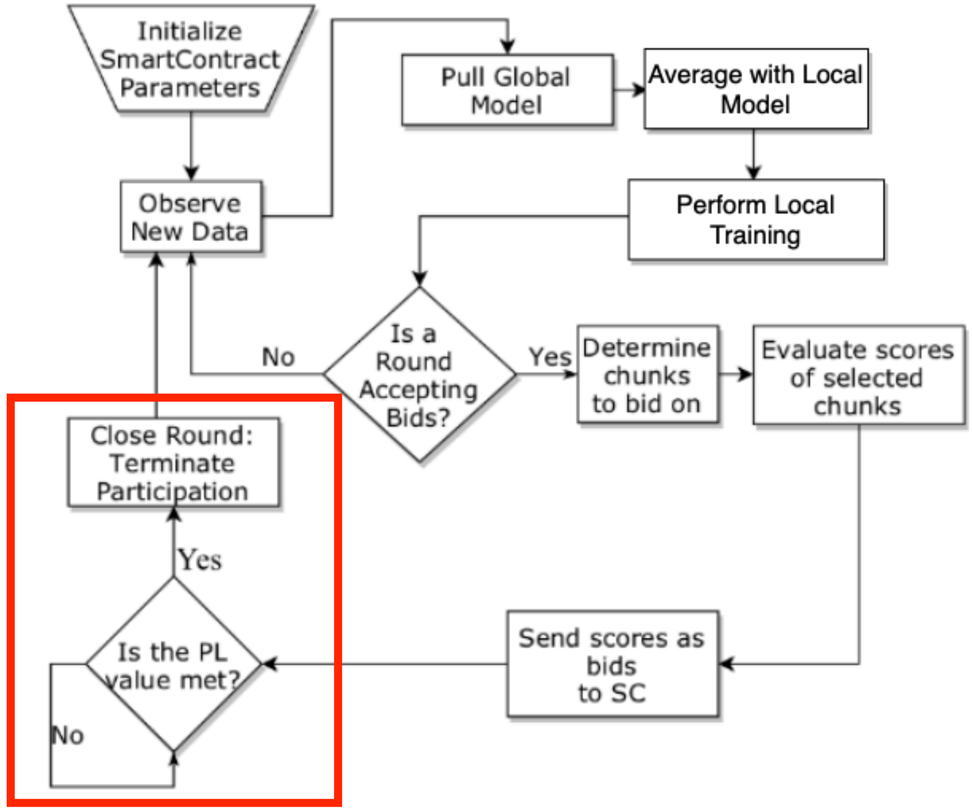


USER ORIENTED LOCAL COMPUTATION

- 라운드의 종료

Algorithm 1 Agent driven SC based update mechanism (on Agent j)

initialize partition scheme $\mathcal{C}$, local model Q_0^j
for $k = 0 \ldots$ **do**
  obtain $Q^c, \forall c \in \mathcal{C}$ from blockchain using SC
  compute $Q_{k+1}^{j,c} \leftarrow \frac{Q^c + Q_k^{j,c}}{2}, \forall c \in \mathcal{C}$
  perform local training and update Q_{k+1}^j
  if round is open for participation **then**
    choose chunks $\tilde{C}^k \subseteq \mathcal{C}, |\tilde{C}^k| = B$ randomly
    calculate scores $\delta_c = ||Q^c - Q_{k+1}^{j,c}||, \forall c \in \tilde{C}^k$
    submit bids $[c, \delta_c], \forall c \in C^k$ to SC
    determine accepted chunk set $C^k \subseteq \tilde{C}^k$
    push C^k to blockchain
  end if
end for



GLOBALLY RELEVANT BLOCKCHAIN DRIVEN FL

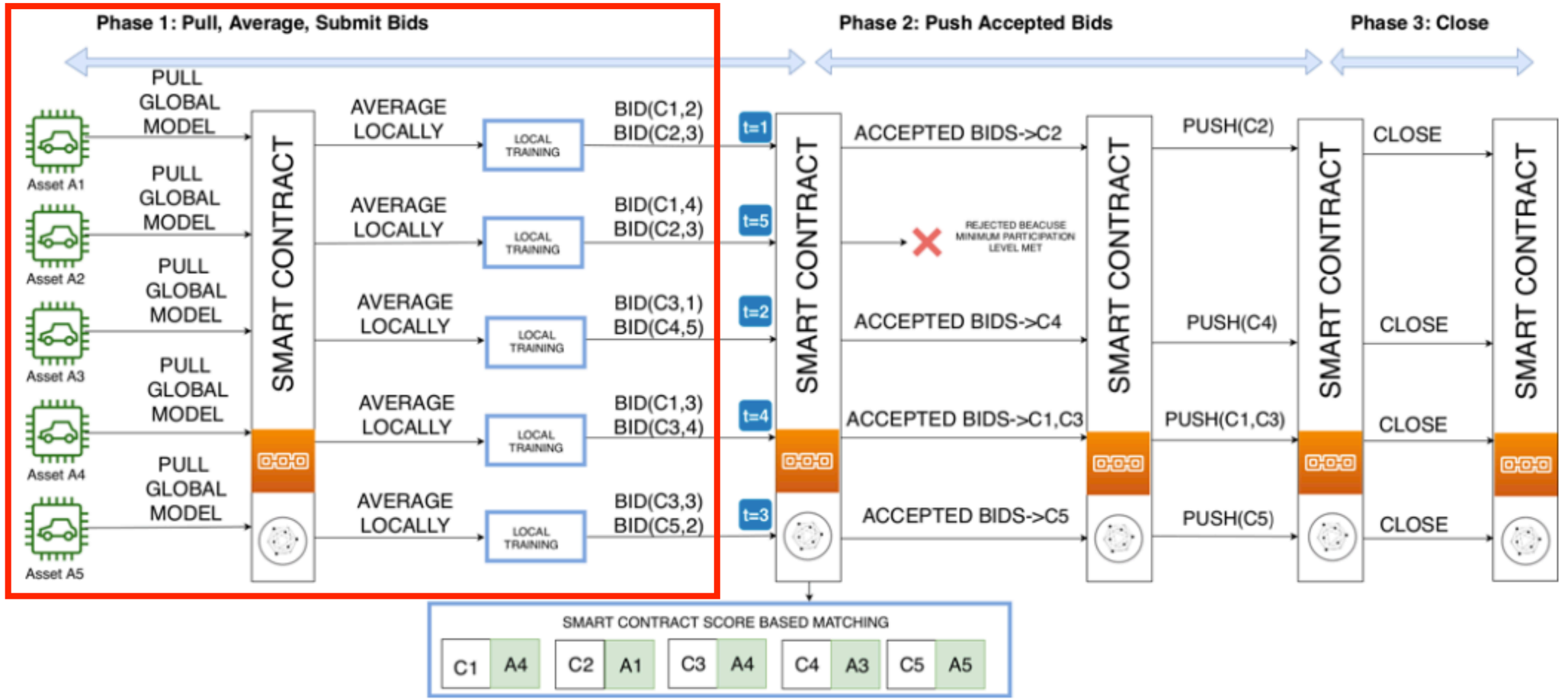
- 글로벌한 관점으로
  - 연산 프로세스는 세 페이즈로 나뉘어짐
    - 1. 가져오기, 평균, 제안 등록
    - 2. 수용된 제안들 입력
    - 3. 폐쇄

GLOBALLY RELEVANT BLOCKCHAIN DRIVEN FL

- 디바이스 A1, ⋯, A5; 모델을 5 청크 C1, ⋯, C5로 나누는 예시
  - PL 값은 4로 가정

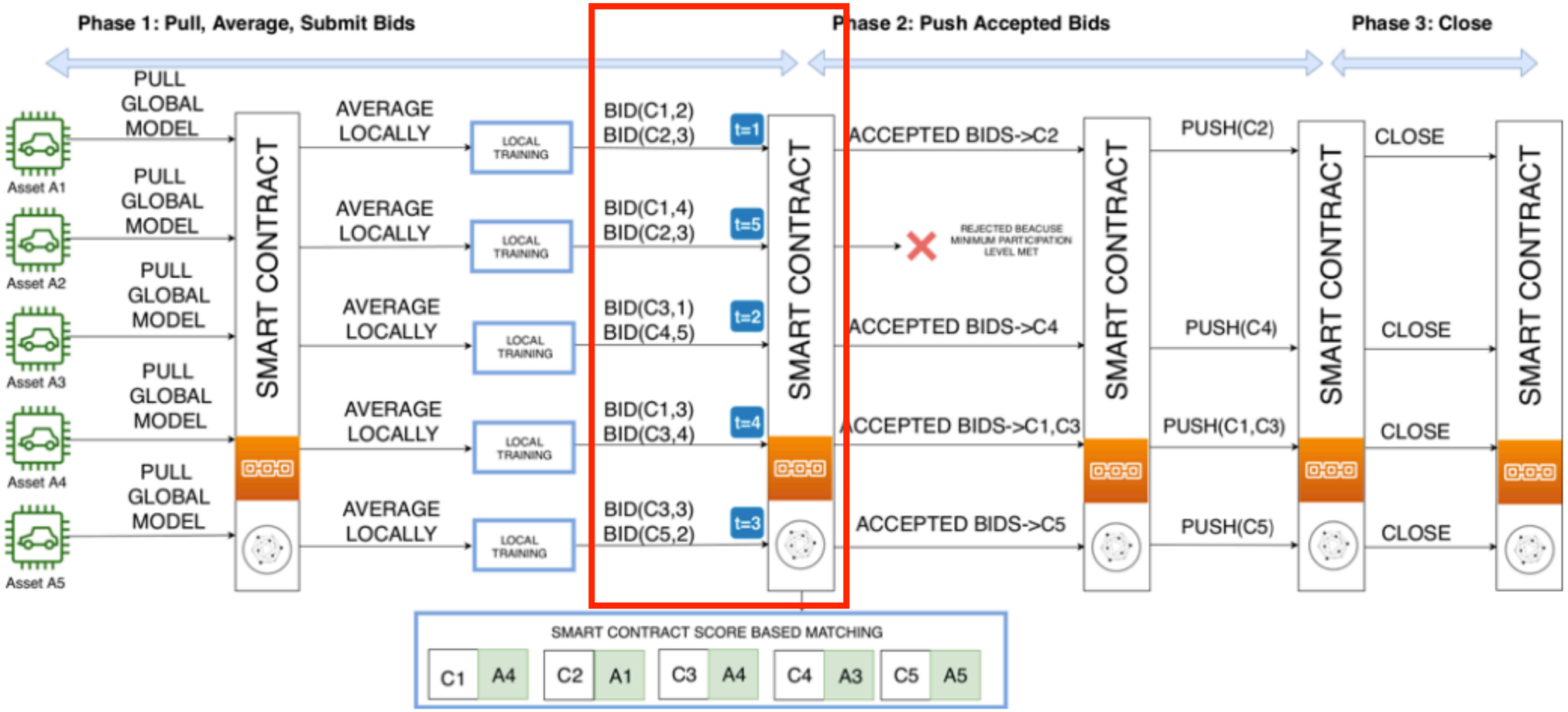
GLOBALLY RELEVANT BLOCKCHAIN DRIVEN FL

- 각 디바이스는 로컬 훈련을 수행한 후 새 제안(들)을 생성



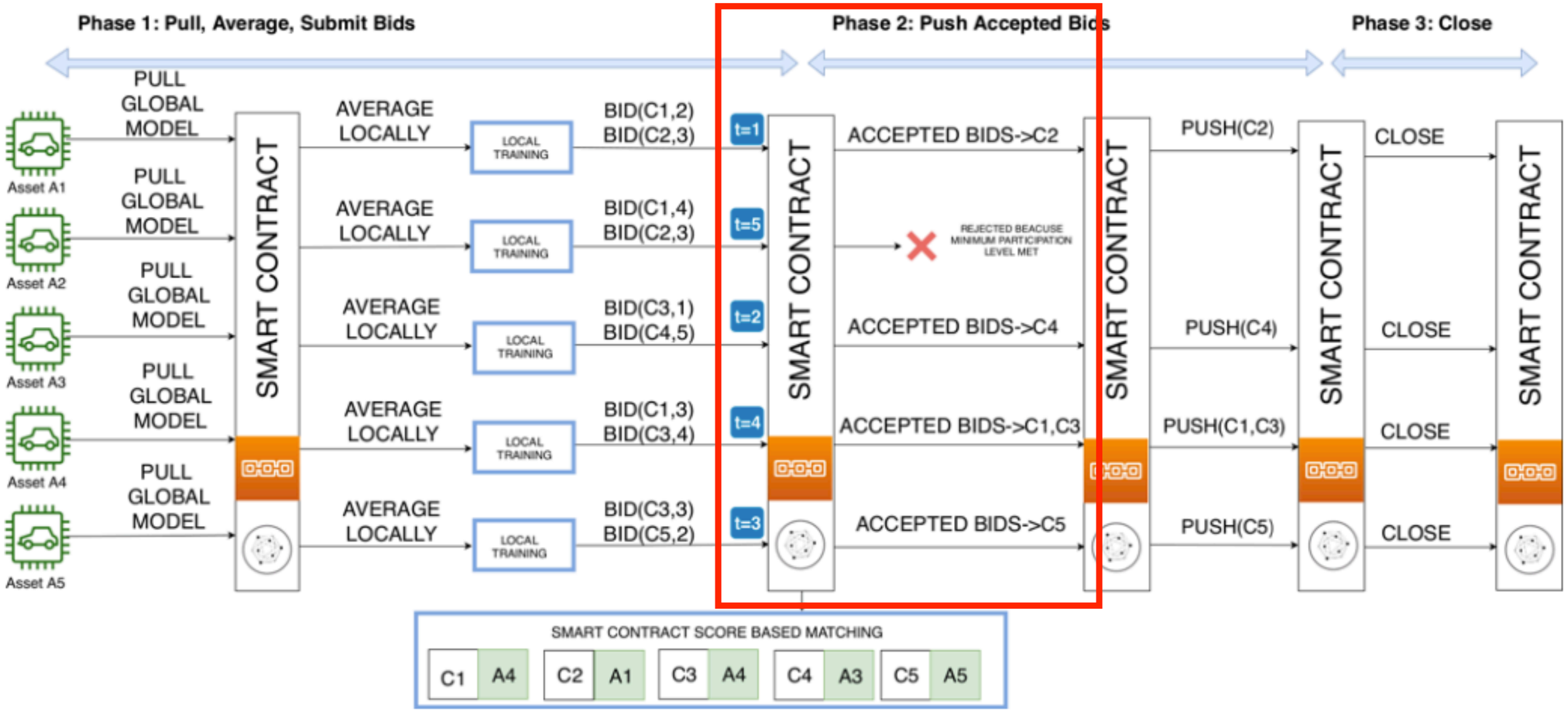
GLOBALLY RELEVANT BLOCKCHAIN DRIVEN FL

- 무작위로 선출한 청크에 대해 제안을 등록



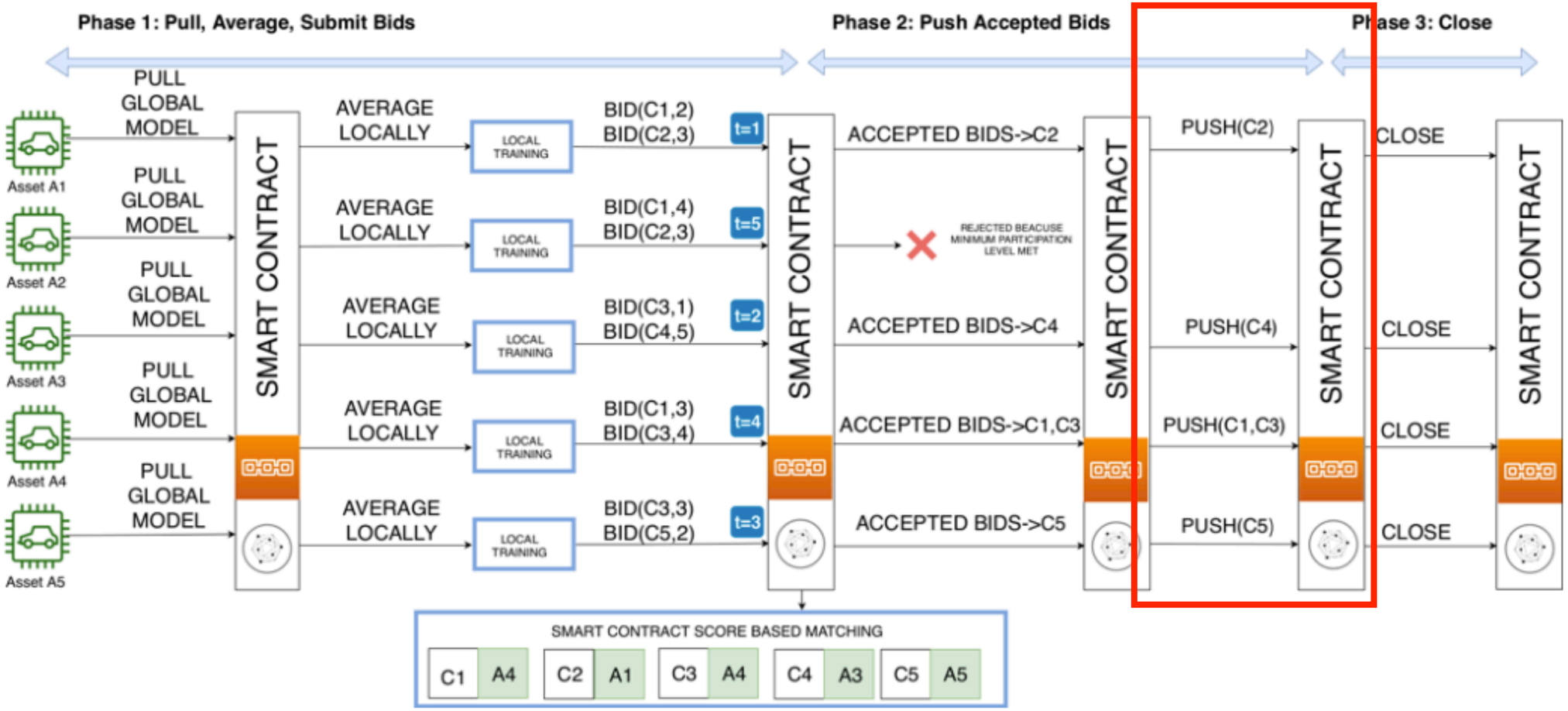
GLOBALLY RELEVANT BLOCKCHAIN DRIVEN FL

- PL 값으로부터 A2의 제안은 거절됨



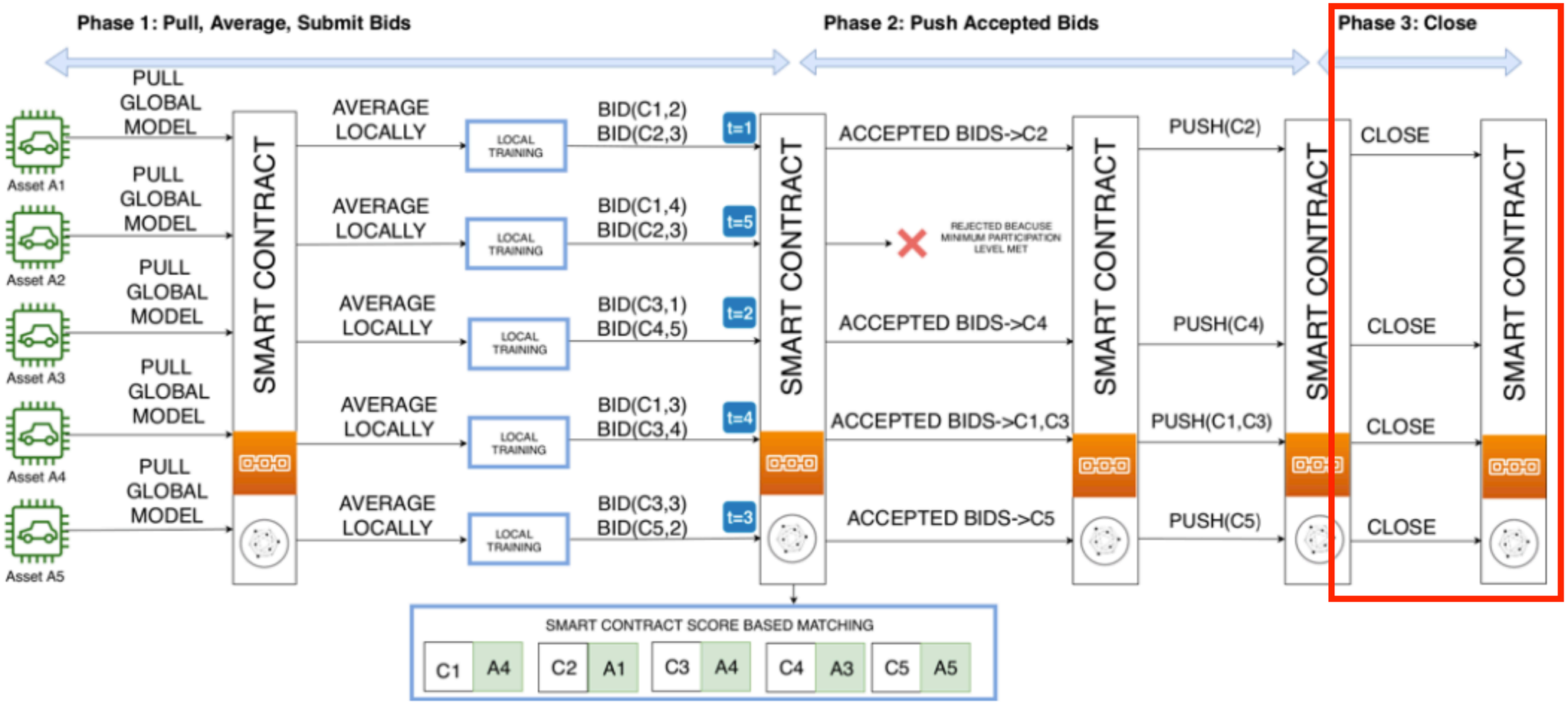
GLOBALLY RELEVANT BLOCKCHAIN DRIVEN FL

- (PL 값으로부터) 수용된 디바이스들은 해당하는 청크들을 등록



GLOBALLY RELEVANT BLOCKCHAIN DRIVEN FL

- 모든 디바이스가 라운드의 종료를 표출



EXPERIMENTS

EXPERIMENTS

- BAFFLE의 효율성을 평가하기 위해 네 가지 핵심 실험을 진행
  - 전통적인 FL과 non-FL과의 비교
  - 청크 수와 사용자 디바이스의 예산에 따른 성능
  - 확장성 분석: 활성 사용자 디바이스 수에 따른 모델 퀄리티 분석
  - PL 파라미터에 따른 강인함 정도

EXPERIMENTS

- 프라이빗 이더리움 블록체인에서 구현 및 실험
  - Geth 사용
  - 16개 노드 사용
  - PoA (Proof-of-Authority)
- 솔리디티로 SC 레이어 구현

EXPERIMENTS

- 프라이빗 블록체인
  - 16코어 32쓰레드 2.40GHz 인텔 제온 CPU
- 사용자 디바이스
  - 12코어 인텔 i7 CPU

EXPERIMENTS

- 2 레이어 DNN
  - 각 레이어에 500 퍼셉트론
  - TensorFlow를 백엔드로 한 Keras로 구현

EXPERIMENTS: BENEFITS STUDY

- Case study: 택시 드라이버 수익
  - 인공지능을 통해 유휴 상태를 줄여 수익 증진
  - 16개의 택시
  - 총 50 라운드 존재
  - 각 라운드마다 택시 당 약 700번의 탑승이 발생함을 가정

EXPERIMENTS: BENEFITS STUDY

- BAFFLE, 로컬 학습(Local Learning, LL), 전통적 FL(CFL) 비교
  - LL은 모델 통합이 없음, 택시도 하나만 가정
  - CFL은 집계자가 존재

EXPERIMENTS: BENEFITS STUDY

- 집계자 free, 탈중앙화 이면서도
  - BAFFLE이 CFL과 거의 유사한 38%의 이익 향상을 이끌어냄
  - LL 대비 18% 이상의 추가 이득

Category	ASR (USD)	Benefit (%)
No Learning (NL)	13387.31	-
Local Learning (LL)	16106.02	20.31
Classical FL (CFL)	18495.94	38.16
BAFFLE	**18442.21**	**37.75**

EXPERIMENTS: SENSITIVITY ANALYSIS

- 청크 사이즈와 예산에 따른 강인함 분석
  - 64개의 택시
  - 총 125 라운드 존재
  - 각 라운드마다 택시 당 약 70번의 탑승이 발생함을 가정

EXPERIMENTS: SENSITIVITY ANALYSIS

- 다양한 청크 사이즈와 예산 크기에 따른 결과

Chunk Size (kB)	No. Of Chunks	Budget Size		
		16	24	32
2	738	38.32	38.18	36.51
4	356	36.37	36.87	39.17
8	181	40.23	34.79	38.11
16	88	39.07	38.82	38.02

  - BAFFLE은
  - 청크 사이즈나 예산과 무관하게
  - 탄력성이 있음

EXPERIMENTS: SCALABILITY ANALYSIS

- 총 활성화 사용자 수 대비 BAFFLE 성능 측정
  - 16, 32, 64, 128개의 택시 상황
  - 총 62 라운드 존재
  - 각 라운드마다 택시 당 약 70번의 탑승이 발생함을 가정

EXPERIMENTS: SCALABILITY ANALYSIS

- 총 활성화 사용자 수 대비 BAFFLE 성능 측정

Taxis	Average ASR (USD)	Benefit (%)
16	14489.59	8.2
32	16547.20	23.6
64	18266.72	36.44
128	18414.48	37.55

  - 사용자 수가 많아지면 모델 퀄리티가 좋아짐
  - 어느 정도 이상의 수준에서는 수렴하게 됨

EXPERIMENTS: PARTICIPATION LEVEL (PL) ANALYSIS

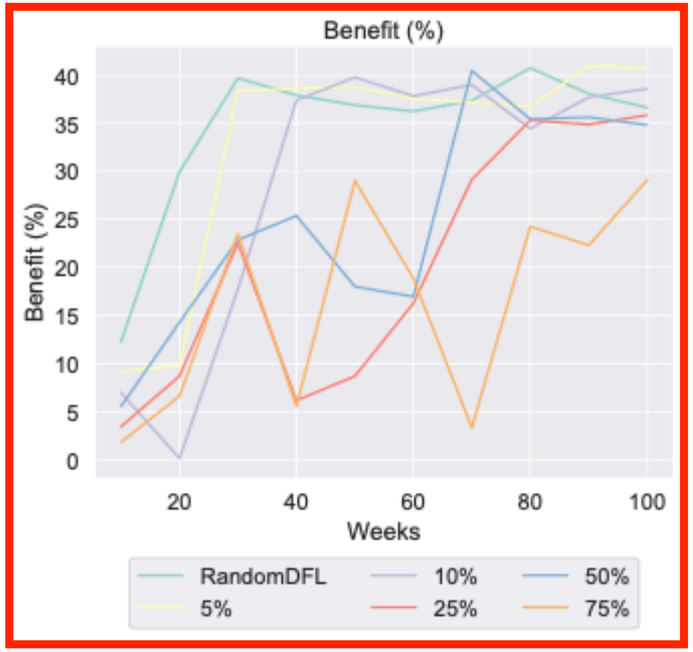
- PL 대비 BAFFLE 성능 측정
  - 64개의 택시
  - 총 62 라운드 존재
  - 각 라운드마다 택시 당 약 70번의 탑승이 발생함을 가정
- PL 값을 (64개의) 5%에서 75% 까지 조정

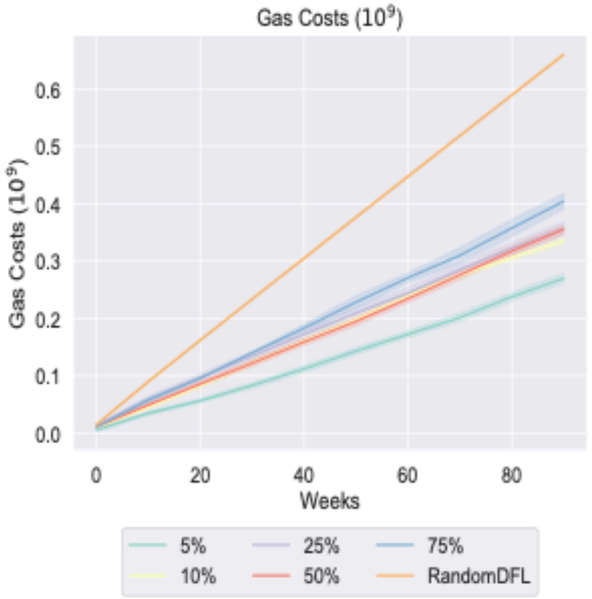
EXPERIMENTS: PARTICIPATION LEVEL (PL) ANALYSIS

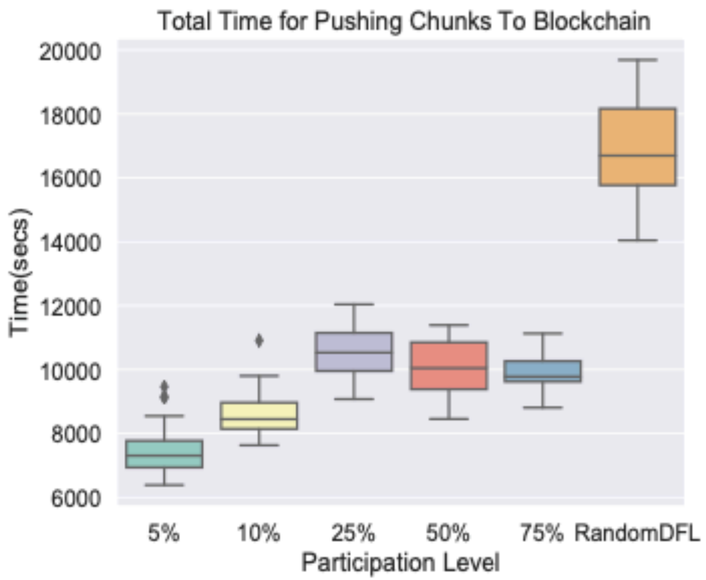
- RandomDFL 방법과 비교
  - R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pp. 1310-1321, ACM, 2015.
  - 단순한(naïve) 집계자 free FL 접근방법
    - PL 값이나 선별 과정 등이 없음

EXPERIMENTS: PARTICIPATION LEVEL (PL) ANALYSIS

- PL에 따른 성능 분석



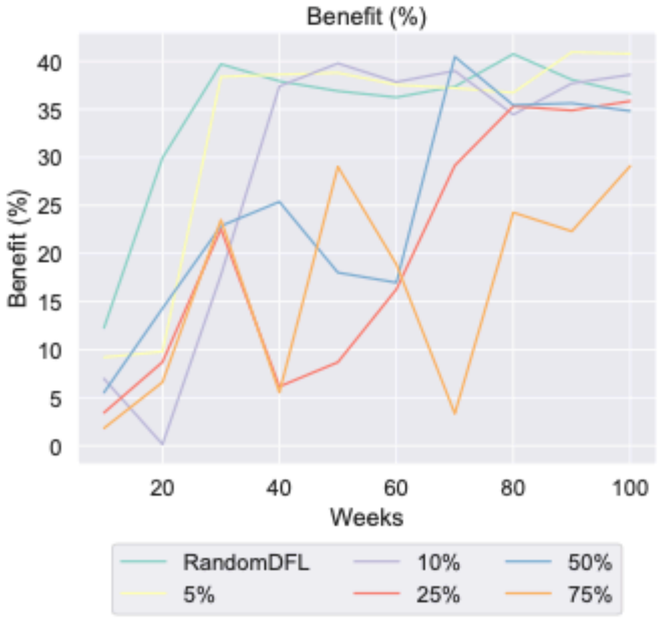


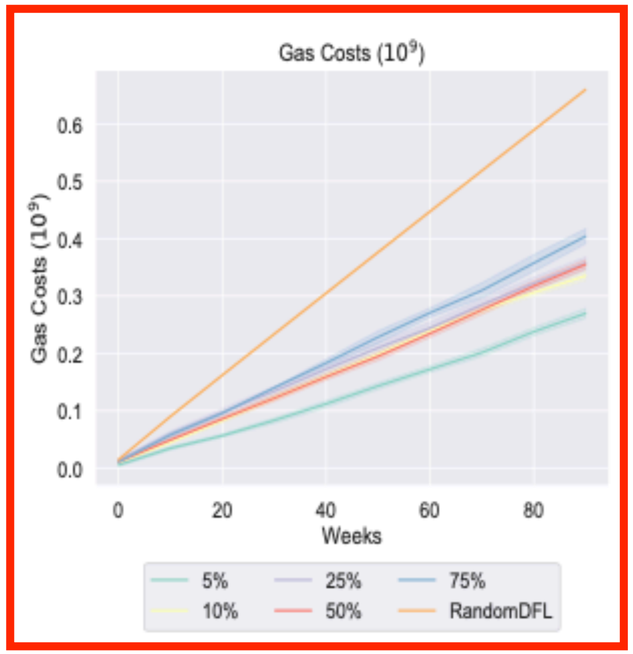


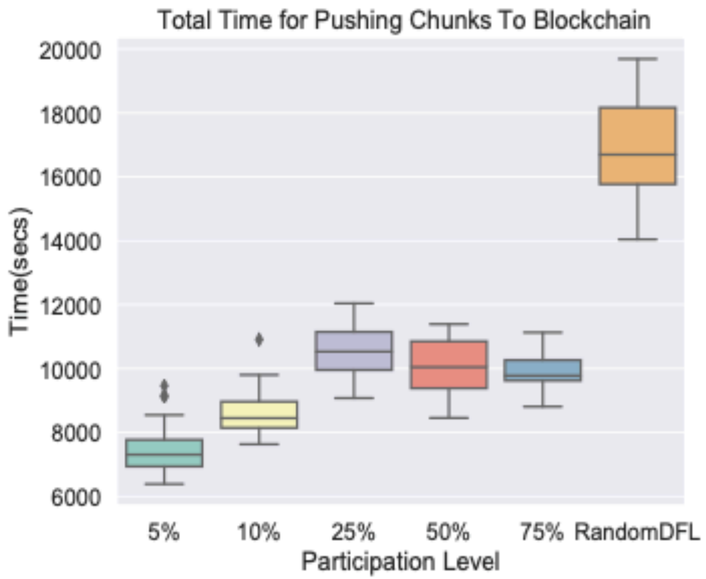
- 모델 퀄리티의 수렴이 가장 빠른 방법은 RandomDFL
- PL이 5% 수준인 BAFFLE도 이와 유사한 성능

EXPERIMENTS: PARTICIPATION LEVEL (PL) ANALYSIS

- PL에 따른 성능 분석



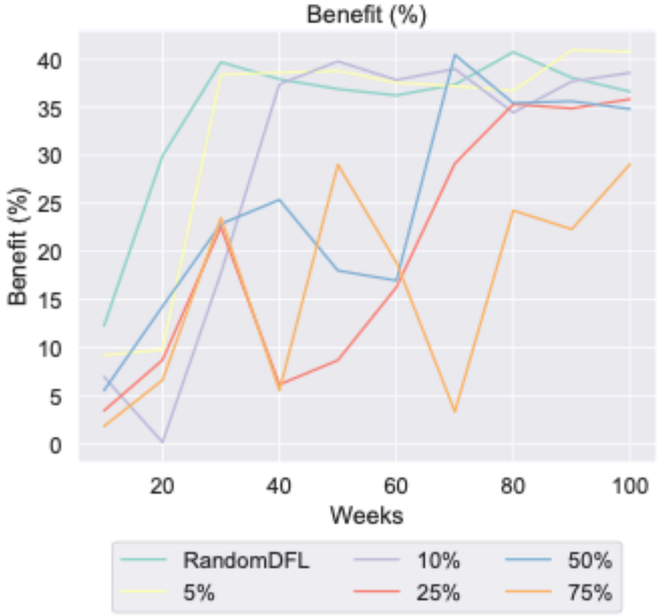


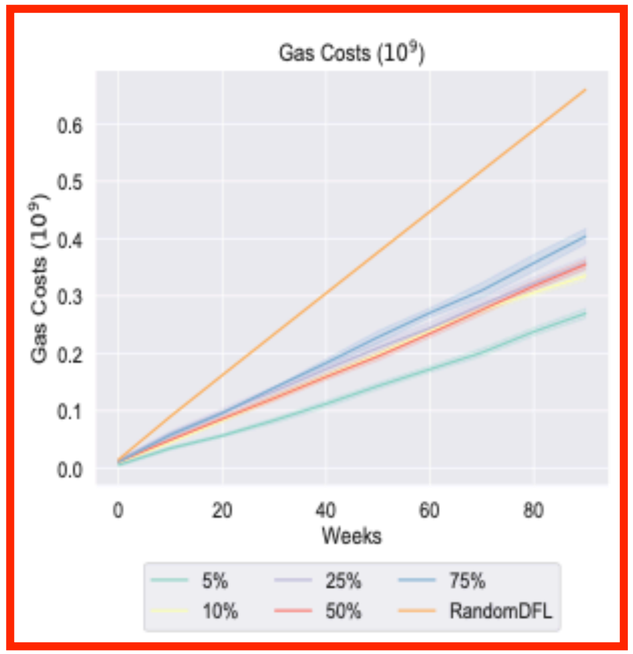


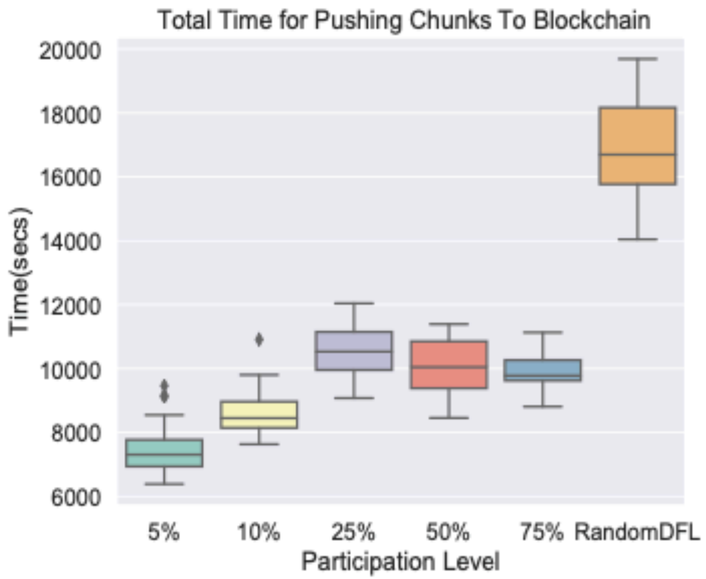
- BAFFLE이 RandomDFL 대비 거의 절반 이하의 Gas 비용만을 소모
  - 불필요한 업데이트를 최소화 하므로

EXPERIMENTS: PARTICIPATION LEVEL (PL) ANALYSIS

- PL에 따른 성능 분석



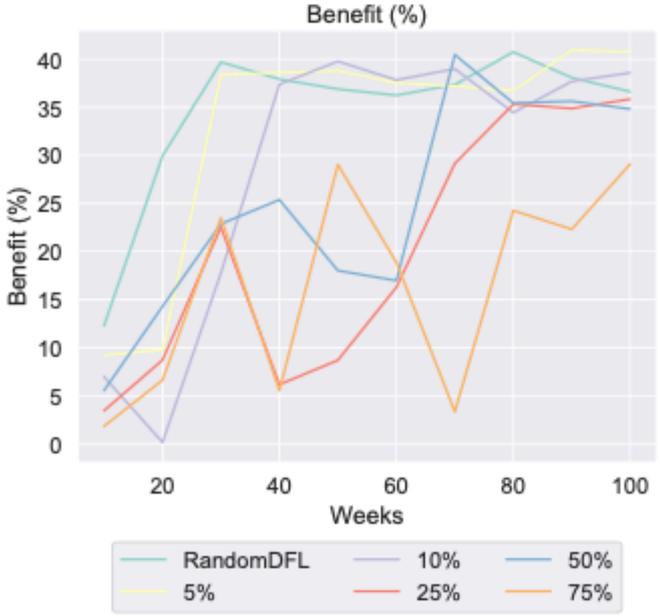


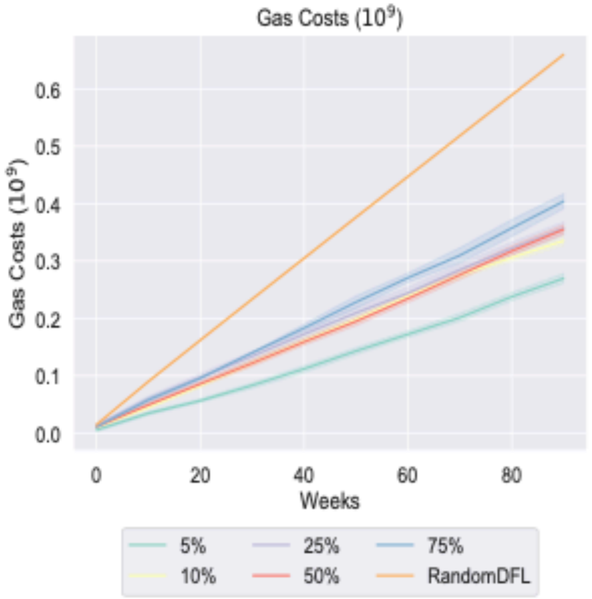


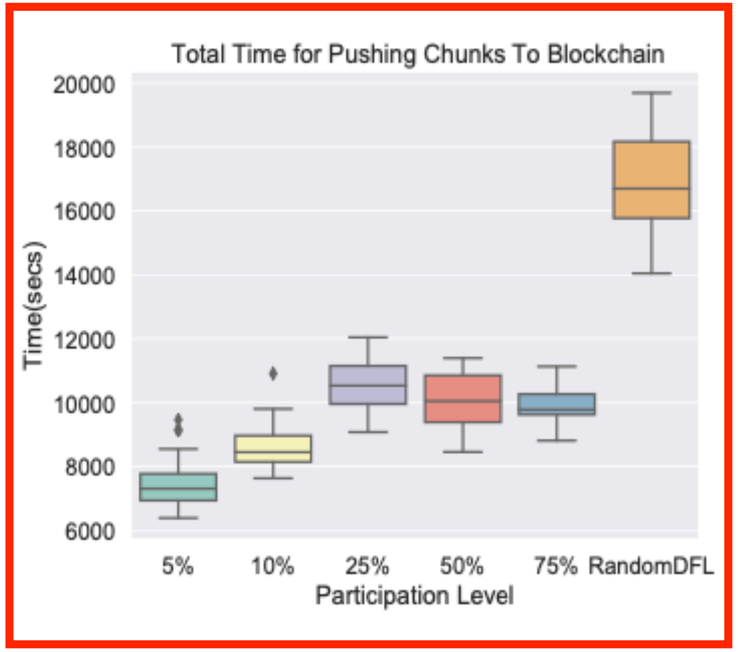
- PL이 5%인 경우 비용 소모가 가장 낮음

EXPERIMENTS: PARTICIPATION LEVEL (PL) ANALYSIS

- PL에 따른 성능 분석







- 디바이스는 각 라운드에서 최고의 청크만을 push하므로 충돌이 없음
  - 가스 비용 및 Push에 소요되는 시간 역시 PL이 5%일 때 가장 낮음

CONCLUSION

CONCLUSION

- BAFFLE
  - 블록체인을 사용해 탈중앙화 집계자 free한
  - FL 메커니즘을 제안 및 구현
- 중앙화된 집계자를 완전히 제거: SC를 통해
  - 라운드의 진행
  - 사용자 디바이스 선택
  - 모델 통합을 수행

CONCLUSION

- 추후 연구
  - 차등 정보보호 (Differential Privacy, DP) 등을 BAFFLE에 도입하는 것
  - CNNs 이나 LSTMs와 같은 복잡한 패러다임을 적용해 보는 것

BAFFLE

BLOCKCHAIN BASED
AGGREGATOR FREE FEDERATED LEARNING