FEDCOIN

A PEER-TO-PEER PAYMENT SYSTEM FOR FEDERATED LEARNING

REFERENCE

- Liu, Yuan, et al.
  "Fedcoin: A peer-to-peer payment system for federated learning."
  Federated Learning. Springer, Cham, 2020. 125-138.

ABSTRACT

ABSTRACT

- 연합 학습
  - Federated Learning (FL)
  - 프라이버시를 지키면서 분산 데이터셋에 대한 학습이 가능
- 데이터 소유자들에 대한 인센티브화를 위해
  - **섀플리 값(Shapley Value, SV)**이 주로 사용됨
  - 그러나 섀플리 값의 계산에는 시간과 비용이 많이 소요됨

ABSTRACT

- FedCoin
  - FL을 위한 블록체인 기반 P2P 지불 시스템
  - SV 기반 수익 분배
- 블록체인 합의 참여자들은
  - SV를 계산
  - Proof-of-Shapley (PoSap) 프로토콜에 의해 새 블록 생성
- Useful Mining으로 볼 수 있음

ABSTRACT

- 계산된 SV에 기반해
  - FL 클라이언트들에게 인센티브가 분배
  - 부인 방지 및 위조 방지 속성도 포함됨

ABSTRACT

- 실제 데이터에 기반한 실험 결과
  - FedCoin이
  - 연산 자원의 상한을 가진 정확히 계산된 SV를 사용해 합의
  - FL 클라이언트들로부터 고품질 데이터(제공)를 증진
- 데이터가 없는 참여자 역시 FL 과정에 참여할 기회가 있음

INTRODUCTION

INTRODUCTION

- FL
  - 각 참여자는 로컬 모델을 훈련
  - 로컬 모델 파라미터로부터 중앙 서버를 업데이트 —> 더 강력한 글로벌 FL 모델
- 중앙화된 ML 방법 대비
  - Raw data 전송에 드는 커뮤니케이션 코스트를 줄이고
  - 서버의 연산 비용도 줄이게 됨
  - 프라이버시 향상

INTRODUCTION

- FL 클라이언트들이 FL 모델에 기여함이 자명함
  - 따라서 FL 커뮤니티를 유지하기 위해서는
  - FL 클라이언트들에게 인센티브를 주는 것이 중요
- 공정하게 다뤄지는 것이 중요
  - FL 모델 기여분에 따라 인센티브가 배분됨이 일반적
  - 가장 많이 사용되는 기여도 측정 방법은 섀플리 값 (SV)

INTRODUCTION

- SV
  - 연합에서 그 기여에 따라 보상을 나눠가지는 잘 알려진 방법
  - 경제, 정보 이론, ML 등 다양한 곳에서 사용되고 있음
- 그러나 SV 계산에는 지수적인 시간이 필요함
  - $O(n!)$: n은 데이터 아이템의 수
- 허용 오차(Marginal error) 하에 SV를 대략적으로 계산하면 연산 비용을 줄일 수 있으나
  - 여전히 비쌈

INTRODUCTION

- FL 시스템이 SV를 계산할 수 있도록 돕기 위해
  - 블록체인 기반 P2P 지불 시스템
  - FedCoin을 제안
- 각 FL 클라이언트의 SV
  - 글로벌 FL 모델에 대한 기여를 의미
  - Proof-of-Shapley(PoSap)로 계산됨

INTRODUCTION

- FedCoin에서 블록체인은
  - 연산 엔진으로 활용되고
  - 지불 기록을 위한 분산 원장으로도 활용됨
- Best of our(저자들의) knowledge,
  - FL에서 인센티브 스킴을 위해 블록체인을 사용하는 최초의 시도라 함
  - Luke's comment: 합의 알고리즘까지 변경한 시도는 많지 않음

RELATED WORK

RELATED WORK: INCENTIVE MECHANISM

- Kang, Jiawen, et al.
  "Incentive mechanism for reliable federated learning:
  A joint optimization approach to combining reputation and contract theory."
  IEEE Internet of Things Journal 6.6 (2019): 10700-10714.
  - 신뢰할 수 없는 데이터 기여자들을 고려한 모델 훈련의 정확도 증진을 위해
    계약 이론(Contract theory)이 사용됨
  - 분산 평판(reputation) 시스템 구축을 위한 컨소시움 블록체인 구조 사용

RELATED WORK: INCENTIVE MECHANISM

- Khan, Latif U., et al.
  "Federated learning for edge networks:
  Resource optimization and incentive mechanism."
  IEEE Communications Magazine 58.10 (2020): 88-93.
  - FL 클라이언트와 FL 서버 양 측의 효용을 최적화하는
    스타켈버그-게임(Stackelberg-game) 기반 인센티브 메커니즘
  - 이기적 FL 클라이언트의 보상 최적화에 초점
  - FL 모델을 이용하기 위해 지불하는 FL 고객이 존재

RELATED WORK: SHAPLEY VALUE

- Li, Yadong, and Xin Cui.
  "Shapley Interpretation and Activation in Neural Networks."
  arXiv preprint arXiv:1909.06143 (2019).
  - ML에서 SV는 널리 사용됨
    - Feature selection
    - 훈련 데이터의 중요도 랭킹 등

RELATED WORK: SECURITY PROBLEMS

- Bao, Xianglin, et al.
  "Flchain: A blockchain for auditable federated learning with trust and incentive."
  2019 5th International Conference on Big Data Computing and Communications (BIGCOM). IEEE, 2019.

- Kim, Hyesung, et al.
  "Blockchained on-device federated learning."
  IEEE Communications Letters 24.6 (2019): 1279-1283.

  - 로컬 모델 파라미터 업데이트를 기록하기 위해 블록체인을 사용

RELATED WORK: TRUST

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

RELATED WORK: TRUST

- Kang, Jiawen, et al.
  "Incentive mechanism for reliable federated learning:
  A joint optimization approach to combining reputation and contract theory."
  IEEE Internet of Things Journal 6.6 (2019): 10700-10714.
  - 블록체인 기반 신뢰 관리 시스템
  - FL 서버가 신뢰도 있고 높은 품질의 데이터 소유자를 선택/선별할 수 있게 함

RELATED WORK

- FedCoin
  - 블록체인을 연산 엔진 및 분산 원장으로 활용하는 것은
  - FL + Blockchain scene에서 최초의 시도

BACKGROUNDS

BACKGROUNDS: SHAPLEY VALUE

- 2012년 노벨 경제학상 공동수상자 중 한 명인 로이드 섀플리 교수가 창시
- 협업에서 각 개인의 기여를 평가하는 방법
- 핵심 아이디어:
  - 함께 일해서 얻은 결과에서 각 개인이 빠진 경우를 계산
  - 그 차이가 곧 기여에 해당
  - 여분의 기여(Marginal contribution)라 함

BACKGROUNDS: SHAPLEY VALUE

- 모든 경우의 수에 대해 기여를 평가하고
  - 참여자 인원수에 따라 기여에 대한 평균을 구함
- 가령 총 3명에 대해
  - k 혼자 일한 경우 기여분 C_1
  - 누구 한 명과 함께 일한 경우의 marginal contribution의 평균이 C_2
  - 누구 두 명과 함께 일한 경우의 marginal contribution의 평균이 C_3 이라면
  - SV는 $(C_1 + C_2 + C_3)/3$으로 계산됨

BACKGROUNDS: P-DISTANCE

- P-거리 (P-distance)
  - $(\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p}$
  - $p = 1$, 맨해튼(Manhattan) 거리
  - $p = 2$, 유클리드(Euclidean 혹은 Euler) 거리
  - $p = infinity$, 체비셰프(Chebychev) 거리

PRELIMINARIES

PRELIMINARIES

- FL 시나리오에서
  - $F_i(w) = l(x_i, y_i; w_t)$
  - 샘플 (x_i, y_i)에 대한 예측의 손실
  - 모델 파라미터 w
  - t 번째 라운드

PRELIMINARIES

- K 클라이언트들이
  - 로컬 데이터셋 D_k를 가짐
  - $n_k = |D_k|$
- 전체 데이터셋 $D = \{D_1, D_2, \ldots, D_k\}$
  - $n = |D| = \Sigma_{k=1}^{K} n_k$

PRELIMINARIES

- 목적 함수는 다음과 같이 최적화되고자 함
  - $$\min_{w \in \mathcal{R}^d} \mathcal{F}(w) \quad \text{where} \quad \mathcal{F}(w) = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in \mathcal{D}_k} \mathcal{F}_i(w)$$
- 이 최적화 문제는
  - Stochastic Gradient Descent(SGD) 기반 방법으로부터
  - 일반적으로 풀 수 있음

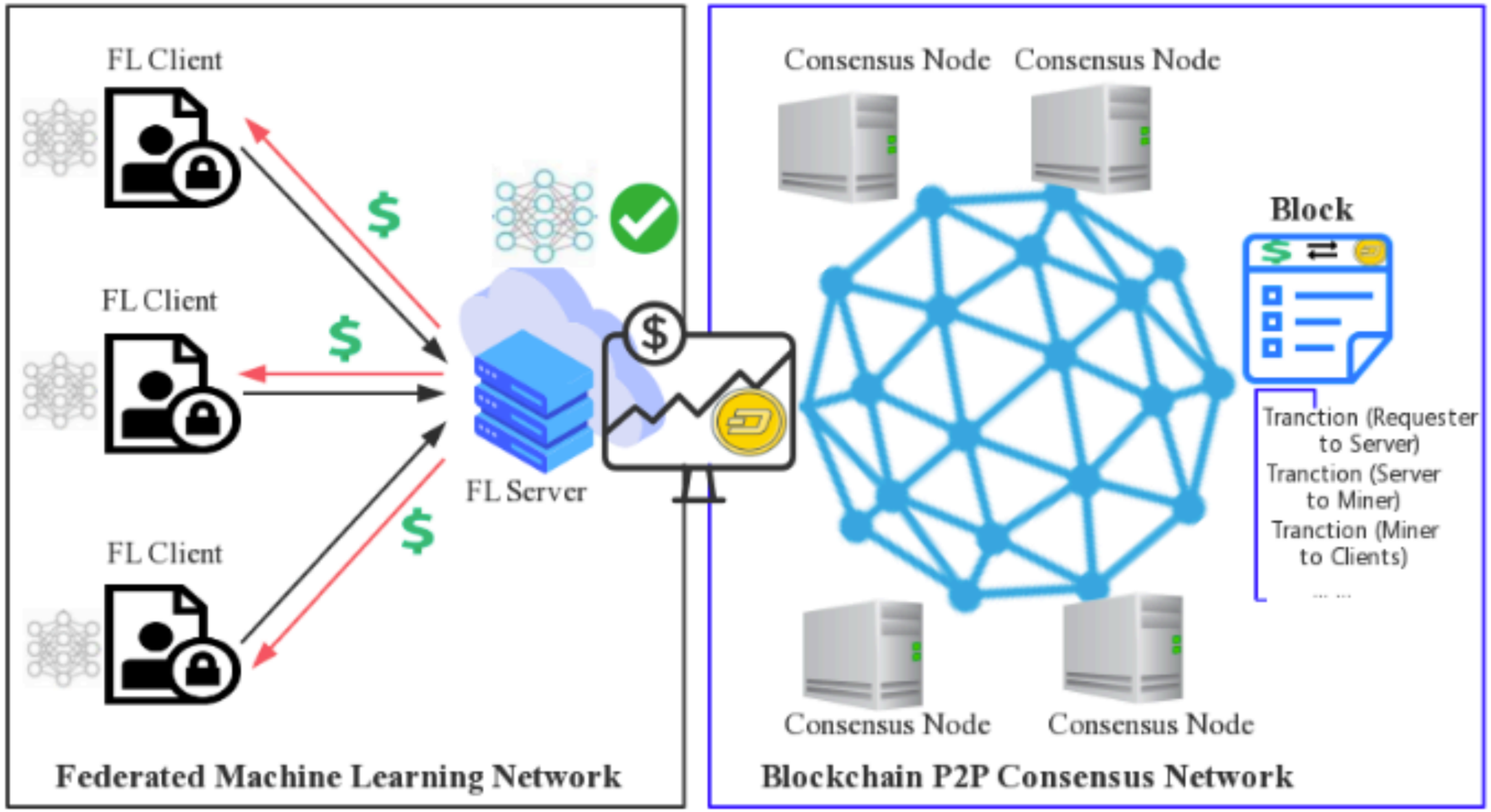
PRELIMINARIES

- 클라이언트 k는
  - 로컬 모델을 그레디언트 g_k^t에 대해
  - $w_k^{t+1} \leftarrow w_t - \eta g_k^t$ 로 업데이트
- FL 서버는 로컬 모델을 통합해 글로벌 FL 모델을 형성
  - $w_{t+1} \leftarrow A(\{w_{t+1}^k = 1,...,K\})$
  - A는 통합 함수 (aggregation function)

FEDCOIN

FEDCOIN

- 두 개의 네트워크가 존재
  - 1) FL 네트워크
  - 2) 블록체인 네트워크



FEDCOIN

- FL 모델 요청자 또는 FL 학습 요청자
  - 예산 V를 가지고 FL 네트워크를 학습시킬 필요가 있는 참여자

FEDCOIN

- FL 서버
  - FL 네트워크에 중앙화된 서버인 FL 서버가 존재
  - 모델 학습을 지도(조정)하고
  - FL 모델 요청자로부터 페이먼트 V(에서 기인한 비용)를 받음

FEDCOIN

- FL 클라이언트
  - 분산 데이터 소유자
  - 공동 학습 업무에 참여하고 페이먼트 V(에서 기인한 비용)를 받음
- 각 FL 클라이언트는
  - 로컬 모델을 학습하고
  - FL 서버에 파라미터 업데이트를 등록

FEDCOIN

- FL 서버의 세 가지 역할
  - 1) 학습 업무를 가격 **TrainPrice**와 함께 FL 클라이언트들에게 공시
  - 2) 보안 통합 프로토콜을 통해 로컬 업데이트를 통합, 연산 페이먼트 **ComPrice**를 획득
    - Bonawitz, Keith, et al.
      ”Practical secure aggregation for privacy-preserving machine learning.” CCS’17.
  - 3) FL 모델을 계산하는 것에 도움을 준 (SV를 계산한)
    블록체인 상의 멤버들에게 수수료 **SapPrice** 지급
- $TrainPrice + ComPrice + SapPrice \leq V$

FEDCOIN

- 각 글로벌 모델 업데이트 이후
  - FL 서버는 FL 클라이언트 각자에 대한 기여도를 계산하기 위한 업무를 공시
- 블록체인 네트워크의 합의 노드
  - SV들을 계산
  - 블록을 채굴한 자(winner, 승자)가 TrainPrice+SapPrice를 획득
  - **ComPrice**를 FL 클라이언트들에게 상응하는 SV에 따라 분배
    - Luke’s comment: TrainPrice인 듯

FEDCOIN

- 현재 디자인에서는
  - 긍정적 기여를 한 클라이언트에게만 보상
  - 부정적 기여에 대한 처벌은 하지 않음

FEDCOIN

- FL 네트워크와 블록체인 네트워크 간 연결
  - 특별한 타입의 업무로부터
- 이 업무는
  - 수신한 로컬 업데이트 집합 $W = \{w_k | k = 1,...,K\}$
  - 통합 함수 A, 손실 함수 $F(w)$
  - 각 업데이트 라운드에 대한 SapPrice와 TrainPrice
  - 를 포함

FEDCOIN

- SapPrice와 TrainPrice는
  - 라운드가 진행됨에 따라 감소함

POSAP

SHAPLEY VALUE

- FL 네트워크로부터 SV 계산 업무를 받으면
  - 블록체인 네트워크의 채굴자들이
  - SV 벡터 $S = [s_k]_{k \in [1,K]}$ 를 계산
  - s_k는 $w_k \in W$를 제출한 클라이언트의 SV에 해당

SHAPLEY VALUE

- 각 채굴자는 독립적으로 SV를 계산
  - 목적은 채굴자의 연산력을 증명하기 위해
  - SV 벡터를 계산하는 것
  - 이를 Proof-of-Shapley(PoSap)이라 함

PROOF-OF-SHAPLEY

- 입력은 FL 네트워크로부터의 업무
- 출력은 새 블록

Algorithm 1: Proof of Shapley (PoSap)

Input: $\mathcal{F}$: Loss function;
$\mathcal{A}$: Aggregation function;
W: Contribution of FL clients in size K;
D: Difficulty in Mining;
Output: Blk: a new block

Initialize $S = [s_k = 0 | k = 1, \ldots, K]$;
time=0;
while *No received* Blk *OR !VerifyBlock(*Blk*)* **do**
  $S_t = [s_k = 0 | k = 1, \ldots, K]$% temporary store S
  Random generate a rank $R = [r_k | k = 1, \ldots, K]$;
  $S_t(R(1)) = \mathcal{F}(\mathcal{A}(W(R(1))))$;
  for i *from* 2 *to* K **do**
    $S_t(R(i)) = \mathcal{F}(\mathcal{A}(W(R(1:i))))$;
    $S_t(R(i)) = S_t(R(i)) - \sum_{j=1}^{i-1} S_t(R(j))$;
  end
  $S = \frac{S \times time + S_t}{time+1}$;
  time=time+1;
  Broadcast S and time;
end
if *Receive a new* S **then**
  Average the Received S to $\overline{S} = \frac{\sum time \times S}{\sum time}$;
  if $\|S - \overline{S}\|_p \leq D$ **then**
    Create a new block Blk after longest chain;
    Broadcast Blk;
    return Blk;
  end
end
if *Receive a new* Blk **then**
  if *VerifyBlock(*Blk*)==ture* **then**
    Update Blk to its chain;
    return Blk;
  end
end

PROOF-OF-SHAPLEY

- 채굴자는 SV 벡터를 0 벡터로 초기화
- 연산 반복 수치(*time*)를 0으로 초기화

Algorithm 1: Proof of Shapley (PoSap)

Input: $\mathcal{F}$: Loss function;
$\mathcal{A}$: Aggregation function;
W: Contribution of FL clients in size K;
D: Difficulty in Mining;
Output: Blk: a new block

Initialize $S = [s_k = 0 | k = 1, \ldots, K]$;
time=0;
while *No received Blk OR !VerifyBlock(Blk)* **do**
  $S_t = [s_k = 0 | k = 1, \ldots, K]$% temporary store S
  Random generate a rank $R = [r_k | k = 1, \ldots, K]$;
  $S_t(R(1)) = \mathcal{F}(\mathcal{A}(W(R(1))))$;
  for *i from* 2 *to* K **do**
    $S_t(R(i)) = \mathcal{F}(\mathcal{A}(W(R(1:i))))$;
    $S_t(R(i)) = S_t(R(i)) - \sum_{j=1}^{i-1} S_t(R(j))$;
  end
  $S = \frac{S \times time + S_t}{time+1}$;
  time=time+1;
  Broadcast S and time;
end
if *Receive a new S* **then**
  Average the Received S to $\overline{S} = \frac{\sum time \times S}{\sum time}$;
  if $\|S - \overline{S}\|_p \leq D$ **then**
    Create a new block Blk after longest chain;
    Broadcast Blk;
    return Blk;
  end
end
if *Receive a new Blk* **then**
  if *VerifyBlock(Blk)==ture* **then**
    Update Blk to its chain;
    return Blk;
  end
end

PROOF-OF-SHAPLEY

- SV 연산은
  다음 두 조건 중 적어도 하나가 만족되는 한 반복됨
  - 1) 새 블록을 받은 적이 없음
  - 2) 수신한 블록이 검증에 실패함

Algorithm 1: Proof of Shapley (PoSap)

Input: $\mathcal{F}$: Loss function;
$\mathcal{A}$: Aggregation function;
W: Contribution of FL clients in size K;
D: Difficulty in Mining;
Output: Blk: a new block

Initialize $S = [s_k = 0 | k = 1, \ldots, K]$;
time=0;
while *No received Blk OR !VerifyBlock(Blk)* **do**
  $S_t = [s_k = 0 | k = 1, \ldots, K]$% temporary store S
  Random generate a rank $R = [r_k | k = 1, \ldots, K]$;
  $S_t(R(1)) = \mathcal{F}(\mathcal{A}(W(R(1))))$;
  for i *from* 2 *to* K **do**
    $S_t(R(i)) = \mathcal{F}(\mathcal{A}(W(R(1:i))))$;
    $S_t(R(i)) = S_t(R(i)) - \sum_{j=1}^{i-1} S_t(R(j))$;
  end
  $S = \frac{S \times time + S_t}{time+1}$;
  time=time+1;
  Broadcast S and time;
end
if *Receive a new S* **then**
  Average the Received S to $\overline{S} = \frac{\sum time \times S}{\sum time}$;
  if $\|S - \overline{S}\|_p \leq D$ **then**
    Create a new block Blk after longest chain;
    Broadcast Blk;
    return Blk;
  end
end
if *Receive a new Blk* **then**
  if *VerifyBlock(Blk)==ture* **then**
    Update Blk to its chain;
    return Blk;
  end
end

PROOF-OF-SHAPLEY

- 이번 반복에서의 연산 결과들을 저장할
  - 임시 벡터 S_t를 생성
- 채굴자가 무작위로
  - K FL 클라이언트들의 랭크를 결정
  - Luke's comment: 순서의 무작위성
- SV를 계산

Algorithm 1: Proof of Shapley (PoSap)

Input: $\mathcal{F}$: Loss function;
$\mathcal{A}$: Aggregation function;
W: Contribution of FL clients in size K;
D: Difficulty in Mining;
Output: Blk: a new block

Initialize $S = [s_k = 0 | k = 1, \ldots, K]$;
time=0;
while *No received Blk OR !VerifyBlock(Blk)* **do**
  $S_t = [s_k = 0 | k = 1, \ldots, K]$% temporary store S
  Random generate a rank $R = [r_k | k = 1, \ldots, K]$;
  $S_t(R(1)) = \mathcal{F}(\mathcal{A}(W(R(1))))$;
  for *i from* 2 *to* K **do**
    $S_t(R(i)) = \mathcal{F}(\mathcal{A}(W(R(1:i))))$;
    $S_t(R(i)) = S_t(R(i)) - \sum_{j=1}^{i-1} S_t(R(j))$;
  end
  $S = \frac{S \times time + S_t}{time+1}$;
  time=time+1;
  Broadcast S and time;
end
if *Receive a new S* **then**
  Average the Received S to $\overline{S} = \frac{\sum time \times S}{\sum time}$;
  if $\|S - \overline{S}\|_p \leq D$ **then**
    Create a new block Blk after longest chain;
    Broadcast Blk;
    return Blk;
  end
end
if *Receive a new Blk* **then**
  if *VerifyBlock(Blk)==ture* **then**
    Update Blk to its chain;
    return Blk;
  end
end

PROOF-OF-SHAPLEY

- 첫 번째 엔티티(entity)에 대해 계산
  - 자기 자신만의
    기여분(가중치)에 대한 손실

Random generate a rank $R = [r_k | k = 1, \ldots, K]$;
$S_t(R(1)) = \mathcal{F}(\mathcal{A}(W(R(1))))$;
for *i from 2 to K* **do**
  $S_t(R(i)) = \mathcal{F}(\mathcal{A}(W(R(1:i))))$;
  $S_t(R(i)) = S_t(R(i)) - \sum_{j=1}^{i-1} S_t(R(j))$;
end

PROOF-OF-SHAPLEY

- 두 번째 엔티티부터 마지막 엔티티까지
  - '여분의 기여'를 계산

Random generate a rank $R = [r_k | k = 1, \ldots, K]$;
$S_t(R(1)) = \mathcal{F}(\mathcal{A}(W(R(1))))$;
for *i from* 2 *to* K **do**
  $S_t(R(i)) = \mathcal{F}(\mathcal{A}(W(R(1:i))))$;
  $S_t(R(i)) = S_t(R(i)) - \sum_{j=1}^{i-1} S_t(R(j))$;
end

PROOF-OF-SHAPLEY

- 1번째부터 i번째 엔티티까지의
  - 가중치를 통합해 손실을 구함
  - 자신을 제외한 기여의 합을 구함
  - 두 값의 차이를 S_t에 저장

Random generate a rank $R = [r_k | k = 1, \ldots, K]$;
$S_t(R(1)) = \mathcal{F}(\mathcal{A}(W(R(1))))$;
for *i from 2 to K* **do**
  $S_t(R(i)) = \mathcal{F}(\mathcal{A}(W(R(1:i))))$;
  $S_t(R(i)) = S_t(R(i)) - \sum_{j=1}^{i-1} S_t(R(j))$;
end

PROOF-OF-SHAPLEY

- 벡터 S는
  - 모든 이전 반복과
  - 임시 벡터 S_t의 평균으로 업데이트됨
- $time$이 1 만큼 증가
- S와 $time$을 전파함

Algorithm 1: Proof of Shapley (PoSap)

Input: $\mathcal{F}$: Loss function;
$\mathcal{A}$: Aggregation function;
W: Contribution of FL clients in size K;
D: Difficulty in Mining;
Output: Blk: a new block

Initialize $S = [s_k = 0 | k = 1, \ldots, K]$;
time=0;
while *No received Blk OR !VerifyBlock(Blk)* **do**
  $S_t = [s_k = 0 | k = 1, \ldots, K]$% temporary store S
  Random generate a rank $R = [r_k | k = 1, \ldots, K]$;
  $S_t(R(1)) = \mathcal{F}(\mathcal{A}(W(R(1))))$;
  for i *from* 2 *to* K **do**
    $S_t(R(i)) = \mathcal{F}(\mathcal{A}(W(R(1:i))))$;
    $S_t(R(i)) = S_t(R(i)) - \sum_{j=1}^{i-1} S_t(R(j))$;
  end
  $S = \frac{S \times time + S_t}{time+1}$;
  time=time+1;
  Broadcast S and time;
end
if *Receive a new S* **then**
  Average the Received S to $\overline{S} = \frac{\sum time \times S}{\sum time}$;
  if $\|S - \overline{S}\|_p \leq D$ **then**
    Create a new block Blk after longest chain;
    Broadcast Blk;
    return Blk;
  end
end
if *Receive a new Blk* **then**
  if *VerifyBlock(Blk)==ture* **then**
    Update Blk to its chain;
    return Blk;
  end
end

PROOF-OF-SHAPLEY

- 채굴자가 S와 $time$을 수신하면
  - 채굴자가 수신한 모든 S에 대해
  - 평균 결과 $\bar{S}$를 계산

Algorithm 1: Proof of Shapley (PoSap)

Input: $\mathcal{F}$: Loss function;
$\mathcal{A}$: Aggregation function;
W: Contribution of FL clients in size K;
D: Difficulty in Mining;
Output: Blk: a new block

Initialize $S = [s_k = 0 | k = 1, \ldots, K]$;
time=0;
while *No received Blk OR !VerifyBlock(Blk)* **do**
  $S_t = [s_k = 0 | k = 1, \ldots, K]$% temporary store S
  Random generate a rank $R = [r_k | k = 1, \ldots, K]$;
  $S_t(R(1)) = \mathcal{F}(\mathcal{A}(W(R(1))))$;
  for i *from* 2 *to* K **do**
    $S_t(R(i)) = \mathcal{F}(\mathcal{A}(W(R(1:i))))$;
    $S_t(R(i)) = S_t(R(i)) - \sum_{j=1}^{i-1} S_t(R(j))$;
  end
  $S = \frac{S \times time + S_t}{time+1}$;
  time=time+1;
  Broadcast S and time;
end
if *Receive a new S* **then**
  Average the Received S to $\overline{S} = \frac{\sum time \times S}{\sum time}$;
  if $\|S - \bar{S}\|_p \leq D$ **then**
    Create a new block Blk after longest chain;
    Broadcast Blk;
    return Blk;
  end
end
if *Receive a new Blk* **then**
  if *VerifyBlock(Blk)==ture* **then**
    Update Blk to its chain;
    return Blk;
  end
end

PROOF-OF-SHAPLEY

- 채굴자는 S와 $\bar{S}$에 대한
  - P-거리를 계산
- 이 거리가
  - 채굴 난이도 D보다 크지 않으면
  - 그 채굴자가 승자(winner)가 됨
  - 새 블록 Blk를 생성
- 난이도 D는 동적으로 변화함

Algorithm 1: Proof of Shapley (PoSap)

Input: $\mathcal{F}$: Loss function;
$\mathcal{A}$: Aggregation function;
W: Contribution of FL clients in size K;
D: Difficulty in Mining;
Output: Blk: a new block

Initialize $S = [s_k = 0 | k = 1, \ldots, K]$;
time=0;
while *No received Blk OR !VerifyBlock(Blk)* **do**
  $S_t = [s_k = 0 | k = 1, \ldots, K]$% temporary store S
  Random generate a rank $R = [r_k | k = 1, \ldots, K]$;
  $S_t(R(1)) = \mathcal{F}(\mathcal{A}(W(R(1))))$;
  for *i from* 2 *to* K **do**
    $S_t(R(i)) = \mathcal{F}(\mathcal{A}(W(R(1:i))))$;
    $S_t(R(i)) = S_t(R(i)) - \sum_{j=1}^{i-1} S_t(R(j))$;
  end
  $S = \frac{S \times time + S_t}{time+1}$;
  time=time+1;
  Broadcast S and time;
end
if *Receive a new S* **then**
  Average the Received S to $\overline{S} = \frac{\sum time \times S}{\sum time}$;
  if $\|S - \overline{S}\|_p \leq D$ **then**
    Create a new block Blk after longest chain;
    Broadcast Blk;
    return Blk;
  end
end
if *Receive a new Blk* **then**
  if *VerifyBlock(Blk)==ture* **then**
    Update Blk to its chain;
    return Blk;
  end
end

PROOF-OF-SHAPLEY

- 채굴자가 새 블록 Blk를 수신하면
  - 이 블록을 검증함
  - 검증에 통과하면 체인을 확장
- 마이닝 프로세스의 종료

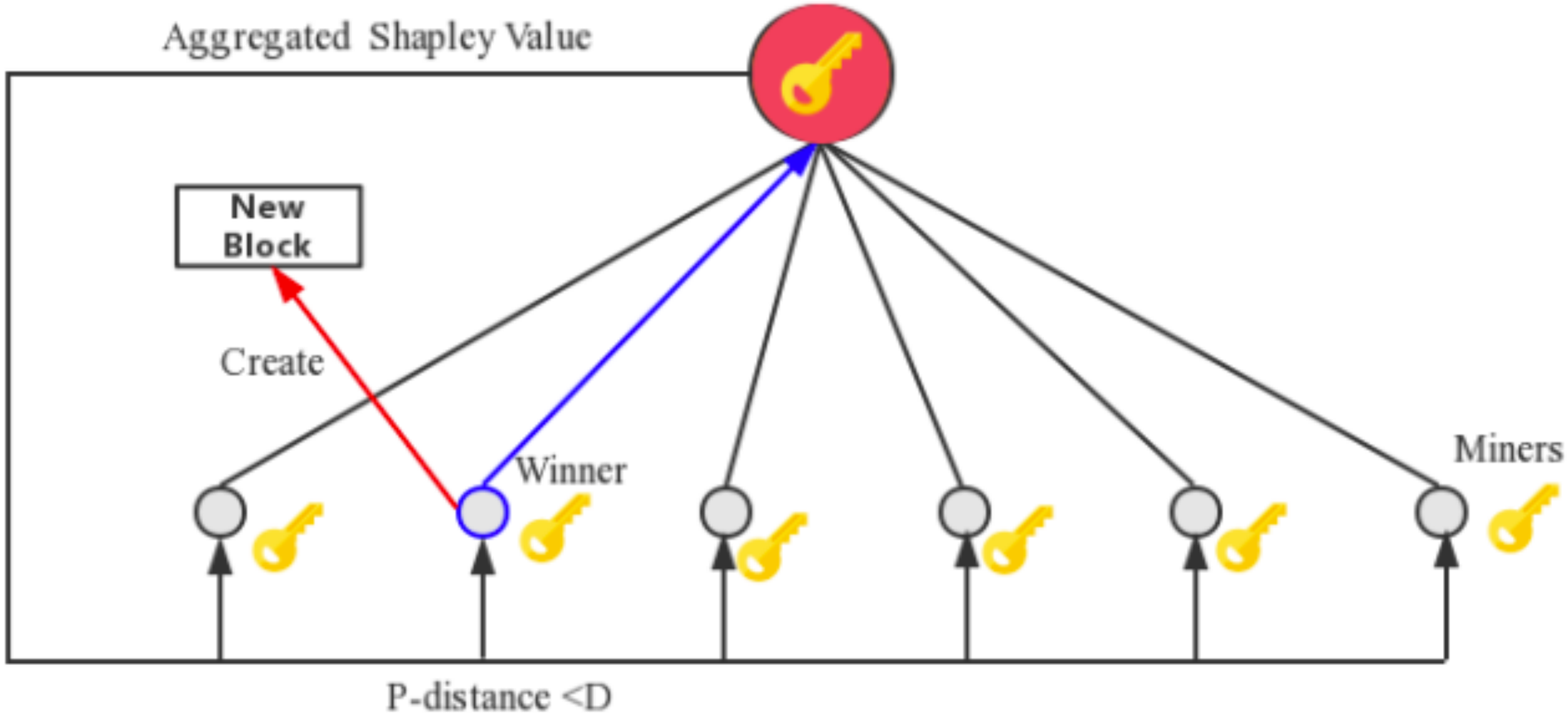
Algorithm 1: Proof of Shapley (PoSap)

Input: $\mathcal{F}$: Loss function;
$\mathcal{A}$: Aggregation function;
W: Contribution of FL clients in size K;
D: Difficulty in Mining;
Output: Blk: a new block

Initialize $S = [s_k = 0 | k = 1, \ldots, K]$;
time=0;
while *No received Blk OR !VerifyBlock(Blk)* **do**
  $S_t = [s_k = 0 | k = 1, \ldots, K]$% temporary store S
  Random generate a rank $R = [r_k | k = 1, \ldots, K]$;
  $S_t(R(1)) = \mathcal{F}(\mathcal{A}(W(R(1))))$;
  for *i from* 2 *to* K **do**
    $S_t(R(i)) = \mathcal{F}(\mathcal{A}(W(R(1:i))))$;
    $S_t(R(i)) = S_t(R(i)) - \sum_{j=1}^{i-1} S_t(R(j))$;
  end
  $S = \frac{S \times time + S_t}{time+1}$;
  time=time+1;
  Broadcast S and time;
end
if *Receive a new S* **then**
  Average the Received S to $\overline{S} = \frac{\sum time \times S}{\sum time}$;
  if $\|S - \overline{S}\|_p \leq D$ **then**
    Create a new block Blk after longest chain;
    Broadcast Blk;
    return Blk;
  end
end
if *Receive a new Blk* **then**
  if *VerifyBlock(Blk)==ture* **then**
    Update Blk to its chain;
    return Blk;
  end
end

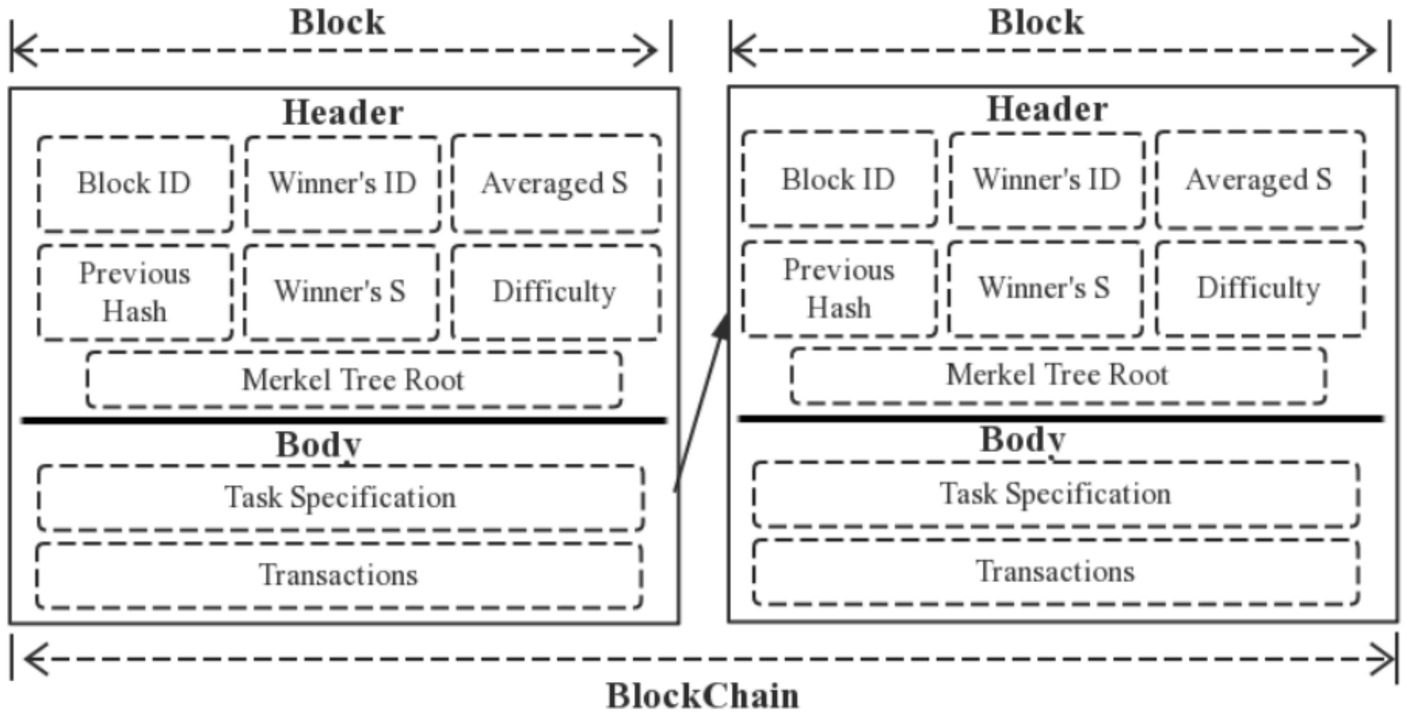
PROOF-OF-SHAPLEY

- S와 $\bar{S}$ 계산에 대한 도식



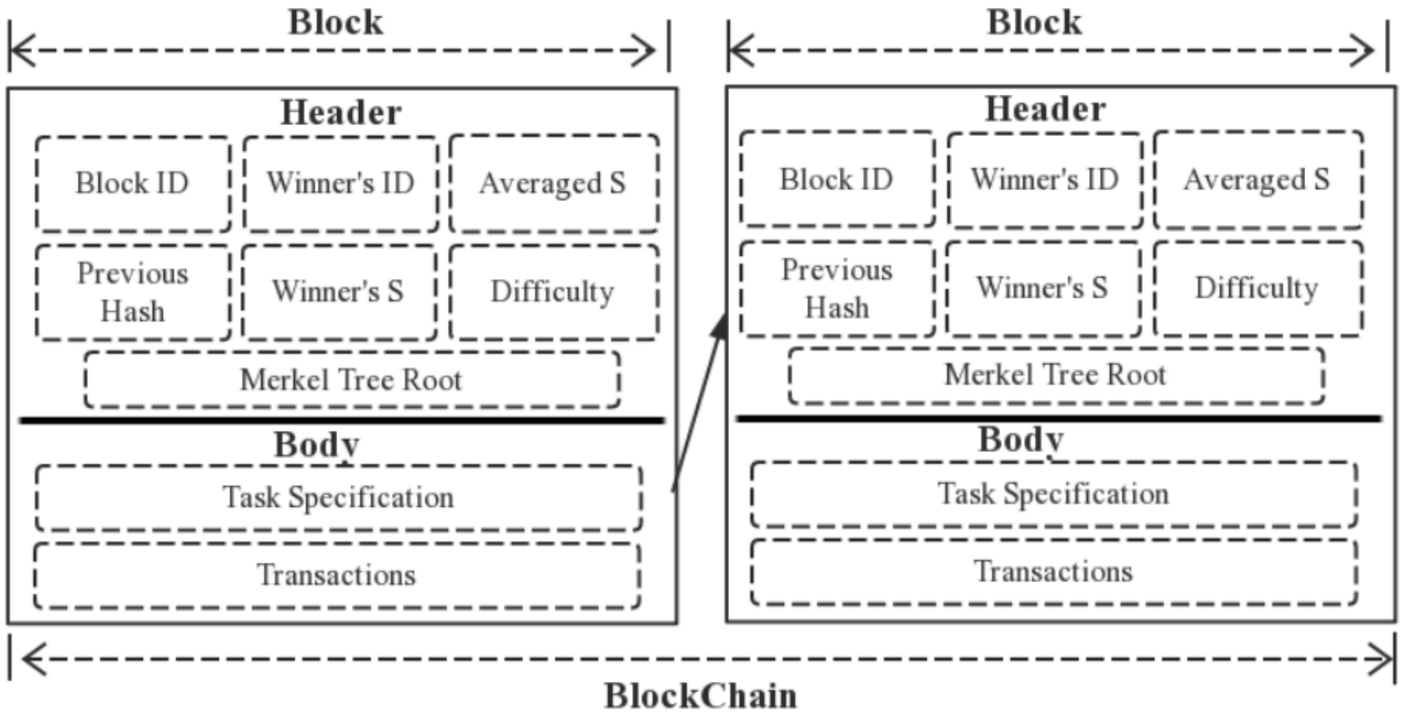
BLOCK STRUCTURE

- Block ID: 블록 높이
- Winner’s ID: 블록 생성자
- Averaged S: $\bar{S}$
- Previous Hash: 이전 블록의 해시
- Winner’s S: 승자의 SV
- Difficulty: 요구 난이도 D
- 머클 트리 루트: 트랜잭션 머클 트리의 루트



BLOCK STRUCTURE

- 블록 바디(body)는 두 종류의 데이터를 기록
  - 업무에 대한 명세
  - 트랜잭션들



BLOCK STRUCTURE

- 채굴자는
  - $\bar{S}$에 따라
  - TrainPrice를 FL 클라이언트들에게 배분

VERIFICATION

- 세 가지 조건이 만족되어야 함
  - $||S_t - \bar{S}_t||_p \leq D$
  - $||\bar{S} - \bar{S}_t||_p \leq D$
  - 현재 블록 ID가 충분히 커야 함 (the longest chain)

VERIFICATION

- 입력은
  - 수신한 새 블록,
  - SV의 로컬 평균,
  - 난이도
- 출력은 True OR False

Algorithm 2: VerifyBlock (new Blk)

Input: Blk: Received new block;
$\overline{S}$: Local average of received Shapley Value;
D: Difficulty in Mining;
Output: ValuationResult: True OR False

```
1  S_t = Blk.S; S_t = Blk.S;
2  if ||S_t − S̄_t||_p ≤ D then
3      if ||S̄ − S̄_t||_p ≤ D then
4          if Blk.ID≥ longest chain length then
5              return ValuationResult=ture;
6          end
7          else
8              return ValuationResult=false;
9          end
10     end
11     else
12         return ValuationResult=false;
13     end
14 end
15 else
16     return ValuationResult=false;
17 end
```

VERIFICATION

- $S_t = Blk\,.\,S$
- $\bar{S}_t = Blk\,.\,\bar{S}$

Algorithm 2: VerifyBlock (new Blk)

Input: Blk: Received new block;
$\bar{S}$: Local average of received Shapley Value;
D: Difficulty in Mining;
Output: ValuationResult: True OR False

```
1  S_t = Blk.S; S̄_t = Blk.S̄;
2  if ||S_t − S_t||_p ≤ D then
3      if ||S̄ − S̄_t||_p ≤ D then
4          if Blk.ID≥ longest chain length then
5              return ValuationResult=ture;
6          end
7          else
8              return ValuationResult=false;
9          end
10     end
11     else
12         return ValuationResult=false;
13     end
14 end
15 else
16     return ValuationResult=false;
17 end
```

VERIFICATION

- $||S_t - \bar{S}_t||_p \leq D$
  - 승자가 올바른 SV 계산 결과로
  - 블록을 생성했는가를 검증

Algorithm 2: VerifyBlock (new Blk)

Input: Blk: Received new block;
$\overline{S}$: Local average of received Shapley Value;
D: Difficulty in Mining;
Output: ValuationResult: True OR False

```
1  S_t = Blk.S; S̄_t = Blk.S̄;
2  if ||S_t − S̄_t||_p ≤ D then
3      if ||S − S_t||_p ≤ D then
4          if Blk.ID≥ longest chain length then
5              return ValuationResult=ture;
6          end
7          else
8              return ValuationResult=false;
9          end
10     end
11     else
12         return ValuationResult=false;
13     end
14 end
15 else
16     return ValuationResult=false;
17 end
```

VERIFICATION

- $||\bar{S} - \bar{S}_t||_p \leq D$
  - 블록의 $\bar{S}$ 값이
  - 로컬 통합 S와 충분히 가까운가
- 비동기 네트워크에서
  - 승자가 충분한 수의 타인의 결과를
  - 수용하도록 요구하는 역할

Algorithm 2: VerifyBlock (new Blk)

Input: Blk: Received new block;
$\overline{S}$: Local average of received Shapley Value;
D: Difficulty in Mining;
Output: ValuationResult: True OR False

```
1  S_t = Blk.S; S̄_t = Blk.S̄;
2  if ||S_t − S̄_t||_p ≤ D then
3      if ||S̄ − S̄_t||_p ≤ D then
4          if Blk.ID≥ longest chain length then
5              return ValuationResult=ture;
6          end
7          else
8              return ValuationResult=false;
9          end
10     end
11     else
12         return ValuationResult=false;
13     end
14 end
15 else
16     return ValuationResult=false;
17 end
```

VERIFICATION

- 블록 ID가 충분히 큰가
  - The longest chain이어야 함
- 포크(Fork)를 효과적으로 피하고
  - 분산 네트워크에서
  - 체인의 안정성을 꾀할 수 있음

Algorithm 2: VerifyBlock (new Blk)

Input: Blk: Received new block;
$\overline{S}$: Local average of received Shapley Value;
D: Difficulty in Mining;
Output: ValuationResult: True OR False

```
1  S_t = Blk.S; S̄_t = Blk.S̄;
2  if ||S_t − S̄_t||_p ≤ D then
3      if ||S̄ − S̄_t||_p < D then
4          if Blk.ID≥ longest chain length then
5              return ValuationResult=ture;
6          end
7          else
8              return ValuationResult=false;
9          end
10     end
11     else
12         return ValuationResult=false;
13     end
14 end
15 else
16     return ValuationResult=false;
17 end
```

DYNAMIC MINING DIFFICULTY

- 새 블록 채굴의 난이도는 동적으로 조정되어야 함
- 난이도 업데이트의 두 요소
  - 1) 채굴자들의 전체 채굴 파워
  - 2) 블록 생성의 속도

PAYMENT SCHEME

PAYMENT SCHEME

- FedCoin 시스템
  - FL 모델 요청자가 V FedCoin을 FL 서버에 예치하면서 시작됨
  - V는 FL 클라이언트들, 블록체인 채굴자들, FL 서버에게 배분됨

PAYMENT SCHEME

- V는 세 부분으로 분할됨
  - TrainPrice: FL 클라이언트들에게 배분됨
  - ComPrice: 모델 통합의 대가로 FL 서버에게 할당됨
  - SapPrice: SV 계산의 대가로 블록체인 네트워크 채굴자들에게 배분됨

PAYMENT SCHEME

- V는 세 부분으로 분할됨
- 분할 비는 사전에 합의된 스마트 컨트랙트로부터 결정됨
  - 가령, TrainPrice:ComPrice:SapPrice=7:1:2

PAYMENT SCHEME

- 입력은
  - 모델 요청자로부터의 V
  - 최종 통합된 SV
- 출력은 V의 배분

Algorithm 3: The Payment Scheme in FedCoin

Input: V: The value paid by a model requester;
$\overline{S}$: The final aggragated Shapley Value;
Output: An allocation of V

while *FL server receives V from a model requester* **do**
  Calculate TrainPrice and SapPrice;
  Publish traing task with price TrainPrice;
  if *The model is well trained* **then**
    Publish a Shapley task to blockchain network with pirce SapPrice;
  end
end
while *a new block is mined* **do**
  FL server transfers TrainPrice+SapPrice to the block winner; **for** *each FL client* i **do**
    if $S_i > 0$ **then**
      $p_i = \frac{S_i}{\sum_{S_j>0} S_j}$ TrainPrice;
      block winner transfers p_i to i;
    end
  end
end

PAYMENT SCHEME

- FL 서버가 모델 요청자로부터 V를 수령하면

Algorithm 3: The Payment Scheme in FedCoin

Input: V: The value paid by a model requester;
$\overline{S}$: The final aggragated Shapley Value;
Output: An allocation of V

```
1  while FL server receives V from a model requester do
2      Calculate TrainPrice and SapPrice;
3      Publish traing task with price TrainPrice;
4      if The model is well trained then
5          Publish a Shapley task to blockchain network
             with pirce SapPrice;
6      end
7  end
8  while a new block is mined do
9      FL server transfers TrainPrice+SapPrice to the block
         winner; for each FL client i do
10         if S_i > 0 then
11             p_i = S_i / (sum_{S_j>0} S_j) TrainPrice;
12             block winner transfers p_i to i;
13         end
14     end
15 end
```

PAYMENT SCHEME

- TrainPrice와 SapPrice를 계산

Algorithm 3: The Payment Scheme in FedCoin

Input: V: The value paid by a model requester;
$\overline{S}$: The final aggragated Shapley Value;
Output: An allocation of V

```
1  while FL server receives V from a model requester do
2      Calculate TrainPrice and SapPrice;
3      Publish traing task with price TrainPrice;
4      if The model is well trained then
5          Publish a Shapley task to blockchain network
             with pirce SapPrice;
6      end
7  end
8  while a new block is mined do
9      FL server transfers TrainPrice+SapPrice to the block
         winner; for each FL client i do
10         if S_i > 0 then
11             p_i = S_i / (sum_{S_j>0} S_j) TrainPrice;
12             block winner transfers p_i to i;
13         end
14     end
15 end
```

PAYMENT SCHEME

- TrainPrice로 훈련 업무를 공시

Algorithm 3: The Payment Scheme in FedCoin

Input: V: The value paid by a model requester;
$\overline{S}$: The final aggragated Shapley Value;
Output: An allocation of V

```
1  while FL server receives V from a model requester do
2      Calculate TrainPrice and SapPrice;
3      Publish traing task with price TrainPrice;
4      if The model is well trained then
5          Publish a Shapley task to blockchain network
             with pirce SapPrice;
6      end
7  end
8  while a new block is mined do
9      FL server transfers TrainPrice+SapPrice to the block
         winner; for each FL client i do
10         if S_i > 0 then
11             p_i = S_i / (sum_{S_j>0} S_j) TrainPrice;
12             block winner transfers p_i to i;
13         end
14     end
15 end
```

PAYMENT SCHEME

- 모델이 잘 훈련되면

Algorithm 3: The Payment Scheme in FedCoin

Input: V: The value paid by a model requester;
$\overline{S}$: The final aggragated Shapley Value;
Output: An allocation of V

```
while FL server receives V from a model requester do
    Calculate TrainPrice and SapPrice;
    Publish traing task with price TrainPrice;
    if The model is well trained then
        Publish a Shapley task to blockchain network
          with pirce SapPrice;
    end
end
while a new block is mined do
    FL server transfers TrainPrice+SapPrice to the block
      winner; for each FL client i do
        if S_i > 0 then
            p_i = S_i / (sum_{S_j>0} S_j) TrainPrice;
            block winner transfers p_i to i;
        end
    end
end
```

PAYMENT SCHEME

- SapPrice로
  섀플리 업무를 블록체인 네트워크에 공시

Algorithm 3: The Payment Scheme in FedCoin

Input: V: The value paid by a model requester;
$\overline{S}$: The final aggragated Shapley Value;
Output: An allocation of V

```
1  while FL server receives V from a model requester do
2      Calculate TrainPrice and SapPrice;
3      Publish traing task with price TrainPrice;
4      if The model is well trained then
5          Publish a Shapley task to blockchain network
             with pirce SapPrice;
6      end
7  end
8  while a new block is mined do
9      FL server transfers TrainPrice+SapPrice to the block
         winner; for each FL client i do
10         if S_i > 0 then
11             p_i = S_i / (sum_{S_j>0} S_j) TrainPrice;
12             block winner transfers p_i to i;
13         end
14     end
15 end
```

PAYMENT SCHEME

- 새 블록이 채굴되면

Algorithm 3: The Payment Scheme in FedCoin

Input: V: The value paid by a model requester;
$\overline{S}$: The final aggragated Shapley Value;
Output: An allocation of V

while *FL server receives* V *from a model requester* **do**
  Calculate TrainPrice and SapPrice;
  Publish traing task with price TrainPrice;
  if *The model is well trained* **then**
    Publish a Shapley task to blockchain network with pirce SapPrice;
  end
end
while *a new block is mined* **do**
  FL server transfers TrainPrice+SapPrice to the block winner; **for** *each FL client* i **do**
    if $S_i > 0$ **then**
      $p_i = \frac{S_i}{\sum_{S_j>0} S_j}$TrainPrice;
      block winner transfers p_i to i;
    end
  end
end

PAYMENT SCHEME

- FL 서버가 TrainPrice+SapPrice를 블록 승자에게 전송
- 각 FL 클라이언트들에 대해 다음을 수행

Algorithm 3: The Payment Scheme in FedCoin

Input: V: The value paid by a model requester;
$\overline{S}$: The final aggragated Shapley Value;
Output: An allocation of V

while *FL server receives V from a model requester* **do**
  Calculate TrainPrice and SapPrice;
  Publish traing task with price TrainPrice;
  if *The model is well trained* **then**
    Publish a Shapley task to blockchain network with pirce SapPrice;
  end
end
while *a new block is mined* **do**
  FL server transfers TrainPrice+SapPrice to the block winner; **for** *each FL client i* **do**
    if $S_i > 0$ **then**
      $p_i = \frac{S_i}{\sum_{S_j>0} S_j}$ TrainPrice;
      block winner transfers p_i to i;
    end
  end
end

PAYMENT SCHEME

- $S_i > 0$ 에 대해
- $p_i = \frac{S_i}{\sum_{S_j>0} S_j} TrainPrice$ 를 전송

Algorithm 3: The Payment Scheme in FedCoin

Input: V: The value paid by a model requester;
$\overline{S}$: The final aggragated Shapley Value;
Output: An allocation of V

```
while FL server receives V from a model requester do
    Calculate TrainPrice and SapPrice;
    Publish traing task with price TrainPrice;
    if The model is well trained then
        Publish a Shapley task to blockchain network
          with pirce SapPrice;
    end
end
while a new block is mined do
    FL server transfers TrainPrice+SapPrice to the block
      winner; for each FL client i do
        if S_i > 0 then
            p_i = S_i / (sum_{S_j>0} S_j) TrainPrice;
            block winner transfers p_i to i;
        end
    end
end
```

EXPERIMENTAL EVALUATION

EXPERIMENTAL EVALUATION

- FedCoin을 평가하기 위해
  - 블록체인 환경을 설정
  - 현실 데이터셋을 이용해 FL을 구성
- FedCoin이 고품질 데이터를 증진시킬 수 있는지
- PoSap의 연산 비용이 현실적인지를 평가

EXPERIMENTAL EVALUATION

- 70,000 이미지의 MNIST
- TensorFlow 사용
- 전통적인 신경망 구조 사용
- FL 통합 함수로는 FedAvg 사용
  - 글로벌 FL 모델 파라미터 계산을 위해 평균을 사용

EXPERIMENTAL EVALUATION

- FL 서버와 100 FL 클라이언트들 설정
  - MNIST 데이터셋은 데이터 품질이 각양각색이 되도록 배분됨
- 10 명의 클라이언트들로 구성된 10개의 그룹
  - 각 그룹의 데이터셋은
  - 10 프리셋 퀄리티 레벨 중 하나를 가짐 (클라이언트의 타입이라 명시)
  - 클라이언트 타입 T_j는 0, 1, ⋯, 9 중에서 균일하게 무작위로 할당됨
- 각 로컬 클라이언트는 모델을 20번씩 훈련

EXPERIMENTAL EVALUATION

- 합의 노드는 도커(Docker)에 기반
- 시뮬레이션 플랫폼
  - CPU: intel i7-7700
  - ? form (CPU Interl i7-7700, GPU 2G, RAM 8g, ROM 1t, SSD 256M). We set $p = 2$ (Euler distance) in PoSap for comparing

EXPERIMENTAL EVALUATION

- P-거리에서 $p = 2$로 설정

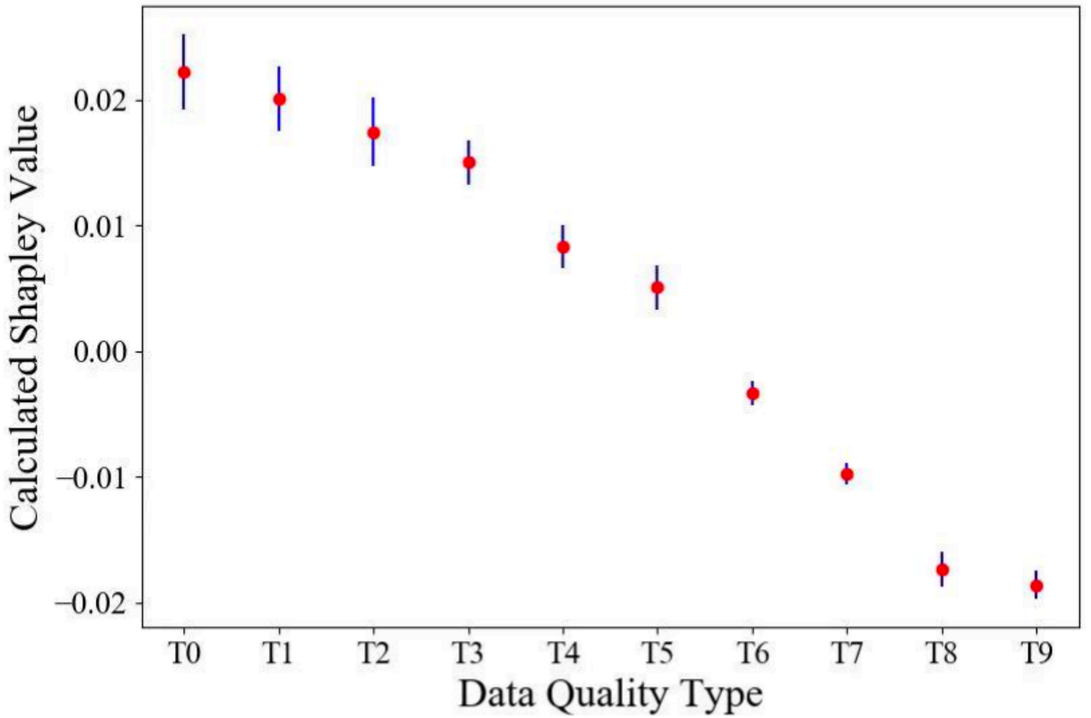
DATA QUALITY EVALUATION

- 데이터 신뢰의 관점에서 데이터 퀄리티 평가
  - Earth Mover's Distance (EMD) 사용
  - 클라이언트의 훈련 데이터 분포와 주어진 분포의 거리를 측정
    - 비교 벤치마크 대상은 전체 MNIST 데이터셋
- 높은 EMD 값은 낮은 품질의 데이터셋임을 의미

Data Type	T_0	T_1	T_2	T_3	T_4
Quality (EMD)	0	0.02	0.04	0.06	0.08
Data Type	T_5	T_6	T_7	T_8	T_9
Quality (EMD)	0.10	0.12	0.14	0.16	0.18

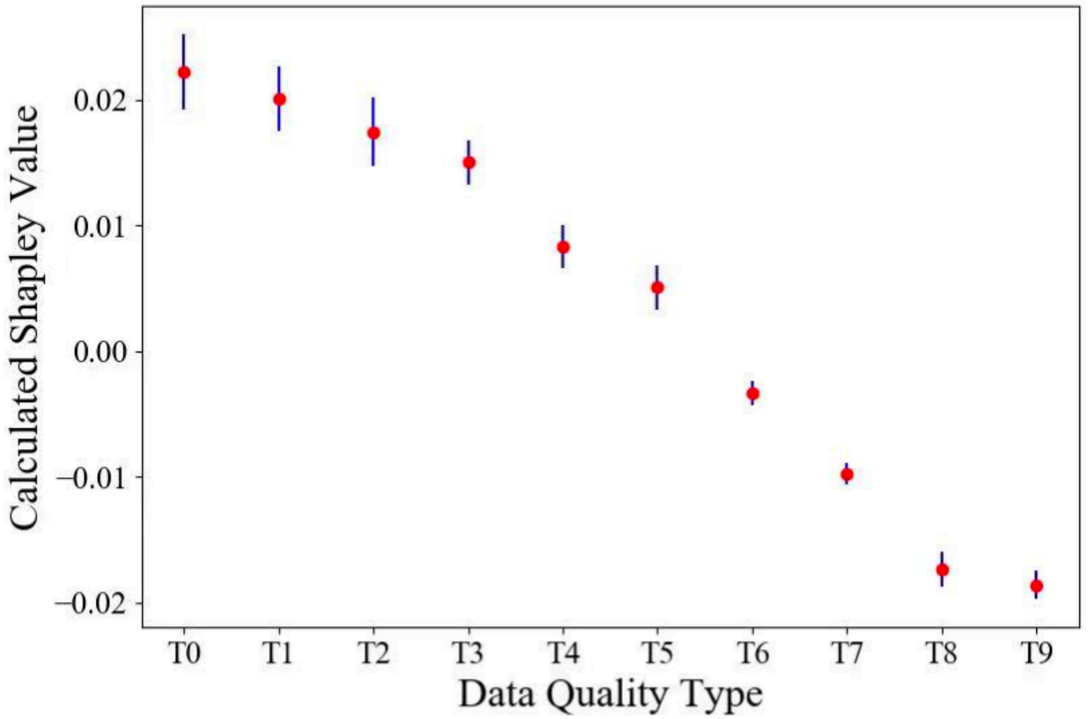
RESULTS AND DISCUSSION

- 평균 SV의 양상
  - 타입 T_0의 클라이언트가 가장 높은 값
  - 타입 T_9의 클라이언트가 가장 낮은 값
  - 품질이 낮아짐에 따라 점차 감소함



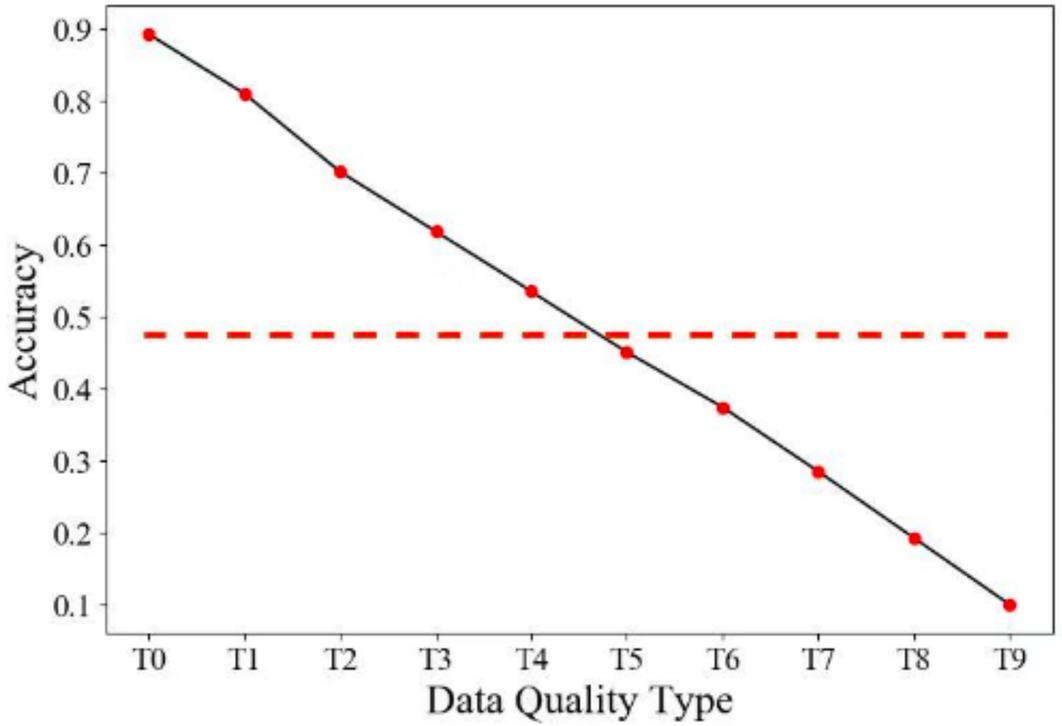
RESULTS AND DISCUSSION

- 평균 SV의 양상
  - 타입 T_0에서 T_4는 양의 값을 가짐
  - 타입 T_5에서 T_9는 음의 값을 가짐
  - 절반의 클라이언트는 양의 기여를 함
  - 나머지 클라이언트는 **음의 기여**를 함
- FedCoin이 고품질 데이터를 증진



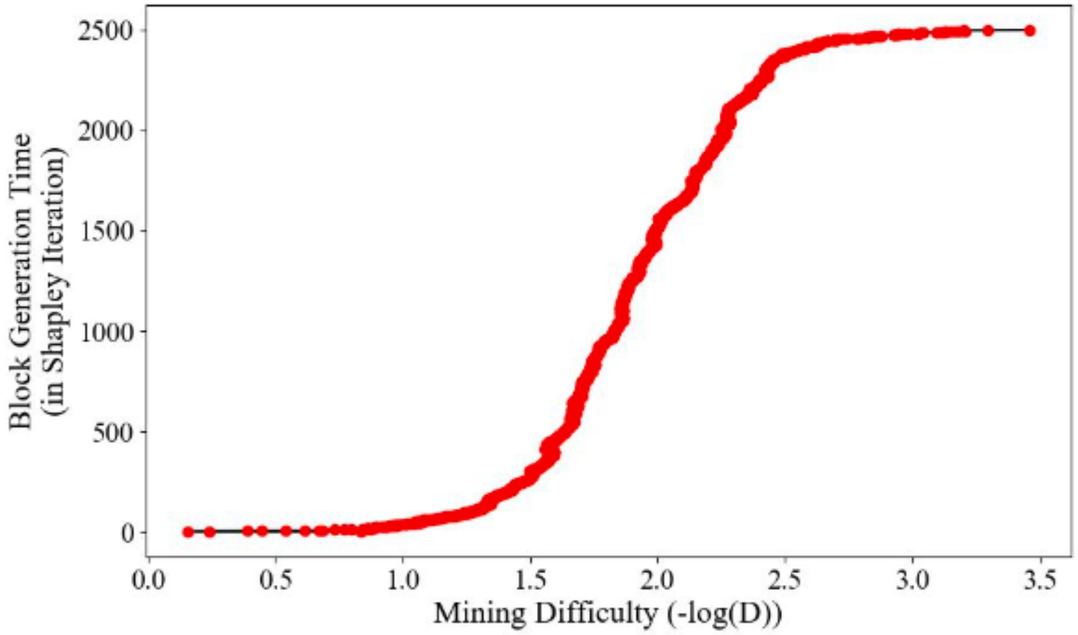
RESULTS AND DISCUSSION

- 왜 음의 SV가 나타나는지에 대한 실험
- 각 클라이언트가 같은 모델을 가지고 학습
  - 모델 정확도가 기준선보다 낮음
  - 기준: 100 클라이언트에 기반한 FL 모델



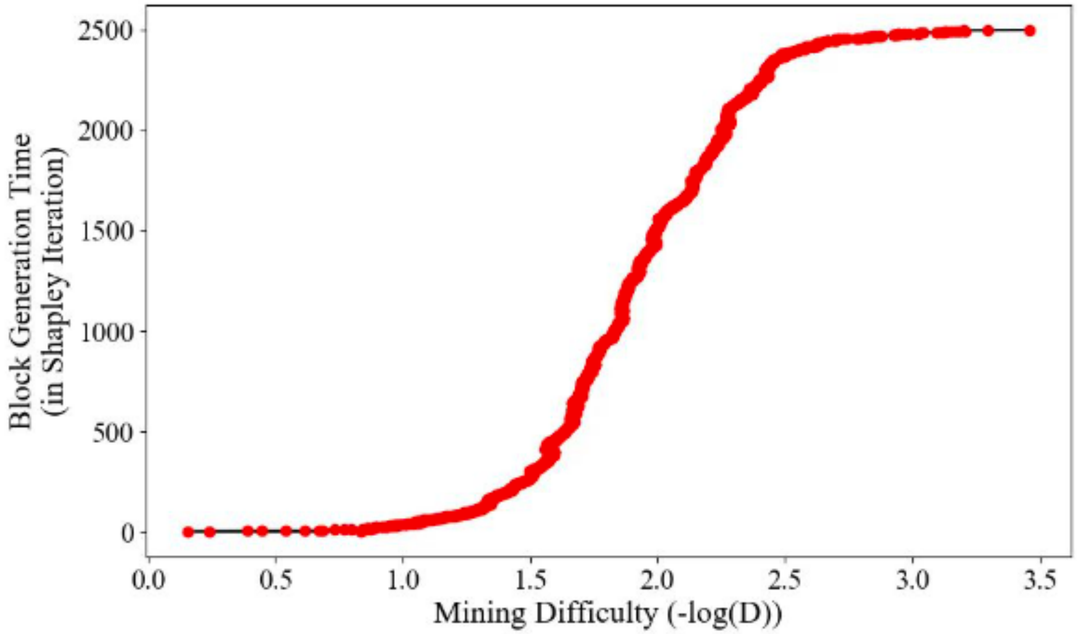
RESULTS AND DISCUSSION

- 블록 생성 시간
  - 샤플리 연산 중 몇 번 반복이 필요한가
  - x축: $-\log D$
    - $x = 2$면 $D = 1e-2$
- 채굴 난이도가 오름에 따라 반복이 더 필요



RESULTS AND DISCUSSION

- 채굴 난이도가 $1e-3$을 넘어서면
  - 약 2,500번 정도의 반복으로 수렴
  - 상한이 존재한다는 뜻
  - 실현성이 높다는 의미



CONCLUSIONS

CONCLUSIONS

- FedCoin
  - 블록체인 기반 연합 학습을 가능하게 하는 페이먼트 시스템
- 각 FL 클라이언트의 SV
  - Proof-of-Shapley (PoSap) 합의 프로토콜으로 계산됨
  - 글로벌 FL 모델에 대한 기여를 의미
- 합의를 이루기 위한 SV 계산에는 상한이 존재함

FEDCOIN

A PEER-TO-PEER PAYMENT SYSTEM FOR FEDERATED LEARNING