2017/04/10, 2017 동계 세미나-블록체인

Storj: A Peer-to-Peer Cloud Storage Network

Shawn Wilkinson et al., 2014

이부형 (boohyung@pel.smuc.ac.kr)

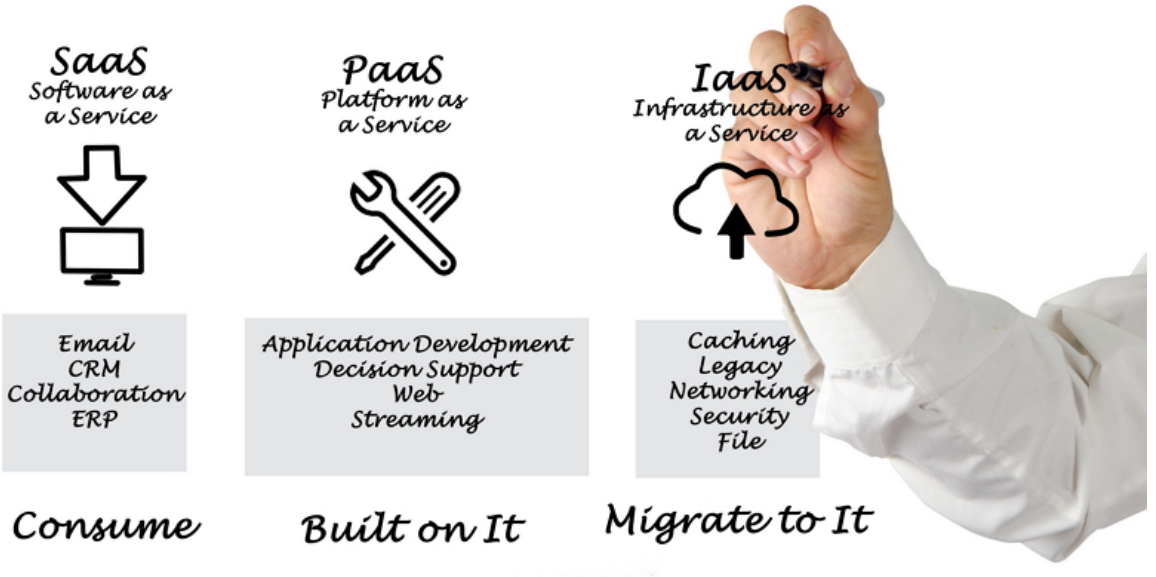
상명대학교 프로토콜공학연구실

Contents

- Introduction
- Main ideas
- Conclusion

Introduction (1/2)

- 클라우드 서비스의 종류
  - Software-as-a-Service (SaaS)
  - Platform-as-a-Service (PaaS)
  - Infrastructure-as-a-Service (IaaS)



Introduction (2/2)

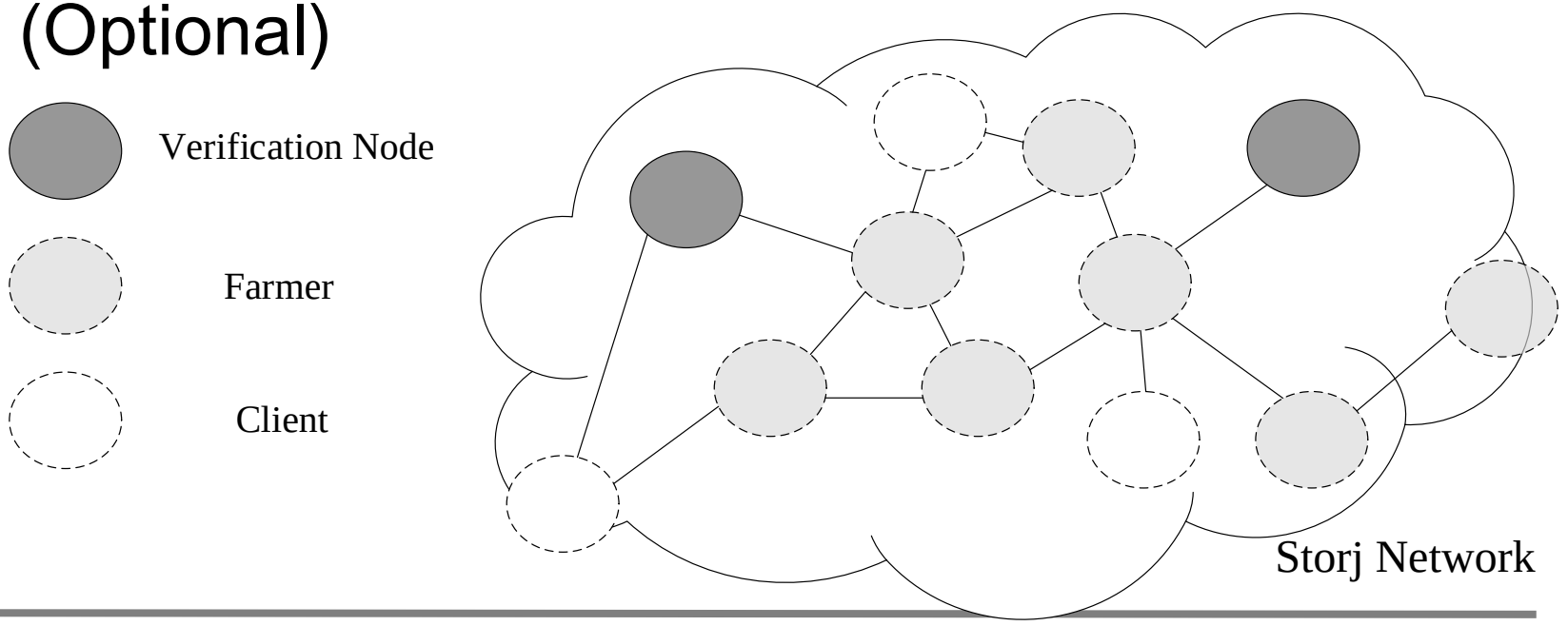
- P2P 기반의 클라우드 서비스: Storj
  - 블록체인 기술을 적용하여 분산 네트워크 구성
    - 서드파티 없이 노드 간 신뢰를 제공
  - 사용자에게 UL/DL와 스토리지를 제공
  - 파일 업로드 및 다운로드, 공유 기능
  - PaaS + IaaS

SERVER BASED NETWORK

PEER TO PEER NETWORK

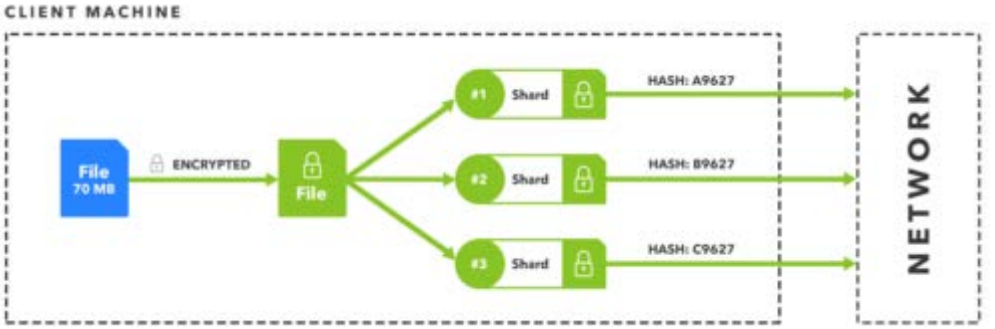
Main ideas (1/10)

- Network Topology
  - Client: 사용자
  - Farmer
    - Client가 업로드한 파일 조각을 저장
    - 저장하는 스토리지를 빌려준 대가로 인센티브를 획득
  - Verification Node: Client가 Offline 상태일 때 auditing을 대리수행 (Optional)



Main ideas (2/10)

- Sharding
  - 파일을 네트워크에 분산 저장하기 위해 여러 조각으로 분할
    - 사용자의 공개키로 암호화/인코딩
    - 암호화된 Shard는 임의의 farmer들에게 전송됨
  - 파일 조각의 단위를 Storj에서는 ‘Shard’ 라고 부름
    - Bittorrent의 Chunk와 같은 개념



Main ideas (3/10)

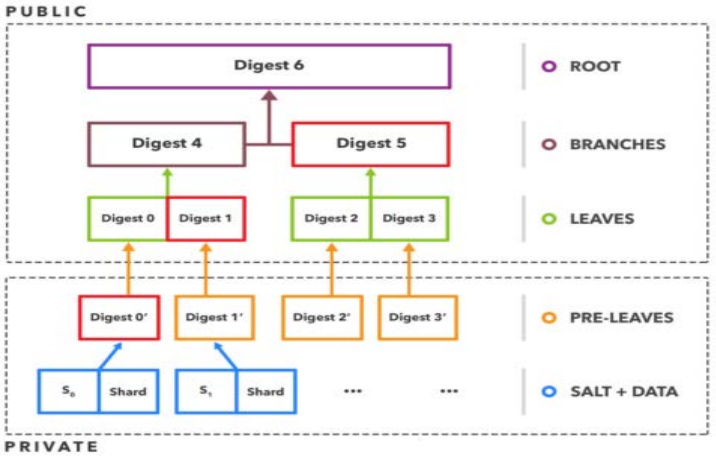
- Redundancy Scheme
  - RAID (Redundant Array of Independent Disk)를 응용하여 파일의 shard를 다수의 farmer에 중복 저장하는 방법
    - 보통 하나의 shard 당 3-4 farmer가 할당되어 저장됨
  - K-of-M Erasure Encoding
    - 전체 M개의 파일 조각 중 최소 K개가 있으면 파일 복구 가능
    - 데이터의 일부가 손실될 경우 디코딩을 통해 복구
  - 파일의 일부가 전송 중 네트워크 오류나 공격자에 의해 삭제되었더라도 백업이 가능함

Main ideas (4/10)

- Proofs of Retrievability
  - 사용자가 자신이 업로드한 데이터의 저장 여부를 확인하기 위한 방법
    - 분산 네트워크의 신뢰성을 보장
  - 머클 트리 기반의 Challenge-response interaction
    - 머클 루트
    - 랜덤한 Salt값을 이용한 해시 코드 생성
  - Storj에서는 ‘auditing’ 혹은 ‘heartbeat’ 이라고 칭함

Main ideas (5/10)

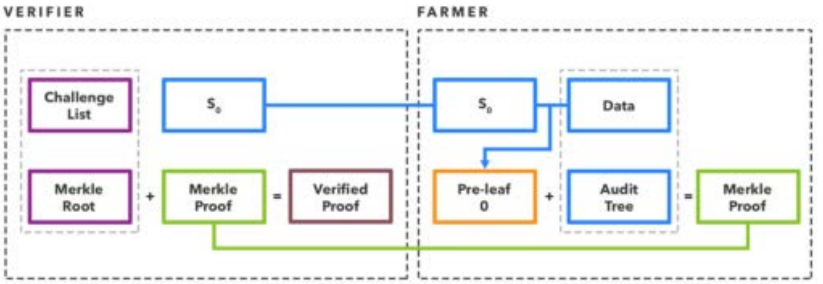
- Proofs of Retrievability
  - 동작 과정
    - Client는 sharding 과정을 마치고 Salt (Seed)를 생성
      - 만약, 파일을 n개의 Shard로 나눴다면 n개의 Salt를 생성, Shard와 각각 연접하여 해시: Pre-leaves를 제작
    - 제작한 pre-leaves는 shard와 함께 다수의 farmer들에게 전송됨
    - 블록의 머클 트리를 구성하는 leaves는 pre-leaves를 한번 더 해시하여 만듬



용어	계산 방법	설명
d_i	Client의 Sharding	$i-th\ Shard$
S_i	Client의 랜덤 값 생성	$i-th\ Salt$
p_i	$H(S_i + d_i)$	$i-th\ Pre-leaves$
l_i	$H(H(S_i + d_i))$	$i-th\ Leaves$

Main ideas (6/10)

- Proofs of Retrievability
  - 검증 과정
    - Client (Verifier): Salt의 집합 S와 머클 루트를 소유
      - $S = \{S_0, S_1, \ldots, S_{n-1}\}$
    - Farmer: Shard와 자신이 저장하고 있는 Shard에 해당하는 pre-leaf를 소유
    - 검증 요청: Client는 S_0와 머클 루트를 farmer에게 전송
      - S_0에 해당하는 pre-leaf를 가지고 있는 farmer
    - 검증 응답: Farmer는 머클 루트를 이용한 검증 결과를 Client에게 전송



Main ideas (7/10)

- Rewards
  - 스토리지/네트워크을 빌려준 대가로 Client는 Farmer들에게 인센티브를 제공
  - 암호화 화폐 Storjcoin X (SJCX)를 이용
  - Client가 auditing 이후 farmer의 SJCX 주소로 인센티브를 전송
  - Farmer의 네트워크 성능(Bandwidth UL/DL), 스토리지 크기에 따라 인센티브를 측정
    - Storage space/Network Bandwidth ∝ Price

Main ideas (8/10)

- How Does it Work? (파일 업로드 및 다운로드)

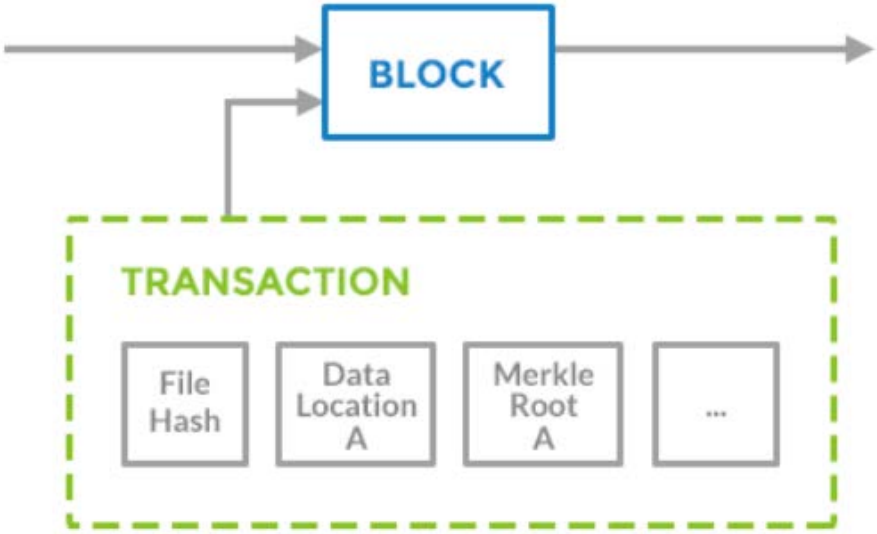
1. Client는 Storj Network에 파일 업로드 요청
2. 파일을 작은 단위로 분할하여 Shard를 생성
   - erasure encoding + encryption
3. 암호화된 Shard를 다수의 farmer들에게 무작위로 배분
4. audit이 완료되면 자신의 shard를 가지고 있는 farmer들에게 대여료를 지급

….

5. Client는 Storj Network에 과거에 업로드한 파일을 다운로드 요청
6. 요청한 파일에 해당하는 shard를 가지고 있는 farmer들은 client에게 자신이 가지고 있는 shard를 전송
7. 암호화된 shard를 복호화하여 조합

Main ideas (9/10)

- Transaction
  - 트랜잭션에는 파일의 메타데이터가 포함됨
    - 필수 항목: 파일의 해시값, Shard의 위치, 머클 루트
  - 트랜잭션에 사용자의 개인키로 디지털 서명
    - 비트코인과 동일
  - 블록은 Client가 auditing 이후에 생성



Main ideas (10/10)

- Security
  - Possible attacks & Defenses
    - Sybil Redundancy Attack/Improper Distribution: 여러 명의 공격자 (farmers)가 서로 공모하여 네트워크를 신뢰하지 못하게 하는 공격 방법
      - 고의로 데이터를 드롭시키거나 동시에 shard의 복사본을 만들어내어 네트워크 트래픽을 조작
      - 방어: Client가 shard를 저장할 farmers를 완전 무작위로 선택한다면 같은 목적으로 공모한 farmer가 선택될 확률은 고려하지 않아도 됨
    - Hostage Byte: Client가 파일을 다운로드할 때, 파일의 마지막 shard를 공격자 (farmer)가 고의로 제공하지 않는 공격
      - 방어: K-of-M erasure encoding으로 공격자로부터 shard를 받지 않고도 파일 획득 (복구)이 가능

Conclusion (1/2)

- P2P 기반 클라우드 스토리지 네트워크 서비스
  - 비트코인 블록체인 기술 + P2P 프로토콜
    - 트랜잭션
    - 공개키 암호화 구조
    - 암호화 해시 함수
    - Kademlia, Distributed Hash Table (DHT) 기반
  - 서드파티에 의존하지 않고 데이터를 전송하고 공유
    - Proof-of-Retrievability
  - 종단간 암호화 구현: 안전한 파일 업로드 및 다운로드
  - 인센티브: 암호화 토큰 사용 (Storjcoin X, SJCX)
- https://storj.io/

Conclusion (2/2)

- Future of Storj
  - Blockchain-as-a-Service (BaaS)
    - 현재 Microsoft Azure, Heroku와 연동하여 서비스
    - 다른 플랫폼과의 연동을 통해 클라우드 스토리지 네트워크 서비스 제공

감사합니다!

이부형 (boohyung@pel.smuc.ac.kr)

백업 1: Erasure encoding

- 스토리지의 복제 및 복구를 위해 사용
  - Erasure Code를 이용하여 데이터를 인코딩
  - 데이터 손실 시 디코딩 과정을 통해 원본 복구
  - 동작 순서
    - 데이터 원본을 n개로 분할
    - n개의 데이터로 인코딩을 통해 k개의 패리티를 생성
    - 데이터 손실 시, n개의 데이터로 디코딩을 통해 원본 데이터를 복구