블록체인 동영상 처리

임 태 희

블록(BLOCK)헤더

- 블록 버전
- 이전 블록의 해시 값
- 머클 루트(Merkle Root): 블록 내 모든 트랜잭션에 대한 해시 값을 가지고 있는 마지막 노드
- 타임스탬프: 블록이 생성된 시간
- 난이도(Difficulty): 블록을 생성하기 위해 필요한 작업 증명(Proof of Work) 난이도
- Nonce: 블록 생성에 사용되는 무작위 숫자 값

블록(BLOCK)트랜젝션

- 블록 내부에 포함된 모든 거래 정보
- 송신자와 수신자의 주소 정보, 전송되는 금액 등

동영상 처리

동영상을 블록 처리하는 방법은 동영상을 작은 블록 단위로 분할하고 블록들을 연결하여 블록체인에 저장하는 것입니다.

1. 동영상을 작은 블록 단위로 분할합니다.
2. 각 블록에는 블록 헤더(Header)와 동영상 데이터가 포함됩니다. 블록 헤더는 블록체인에서 다른 블록들과 연결되는 정보를 담고 있습니다.
3. 동영상의 첫 번째 블록은 이전 블록의 해시 값이 없으므로, 최초 블록인 제네시스 블록(Genesis Block)으로 설정합니다.
4. 이후 생성되는 블록들은 이전 블록의 해시 값을 참조하여 생성됩니다.
5. 동영상의 모든 블록들이 생성되면, 이를 블록체인에 추가합니다.
6. 블록체인의 모든 노드들은 동영상 블록체인을 복제하고 검증하여 동영상의 무결성을 보장합니다.

동영상 블록처리 예제 (PYTHON)

```python
if __name__ == '__main__':
    blockchain = Blockchain()
    for i, block_data in enumerate(read_video_file('movie.mp4')):
        blockchain.add_block(block_data)
        print(f"Block {i+1} has been added to the blockchain.")
```

```python
BLOCK_SIZE = 1024 * 1024  # 1MB


def hash_block(block):
    """블록의 내용을 해싱해서 해시값을 반환합니다."""
    sha = hashlib.sha256()
    sha.update(block.encode())
    return sha.hexdigest()


def read_video_file(filename):
    """동영상 파일을 블록 단위로 나누어서 반환합니다."""
    with open(filename, 'rb') as f:
        while True:
            block = f.read(BLOCK_SIZE)
            if not block:
                break
            yield block
```

동영상 블록처리 예제 (PYTHON)

```python
class Block:
    """동영상 파일을 나타내는 블록 클래스입니다."""

    def __init__(self, index, data, previous_hash):
        self.index = index
        self.data = data
        self.previous_hash = previous_hash
        self.hash = self.calculate_hash()

    def calculate_hash(self):
        """블록의 해시값을 계산합니다."""
        sha = hashlib.sha256()
        sha.update(str(self.index).encode())
        sha.update(self.data)
        sha.update(self.previous_hash.encode())
        return sha.hexdigest()
```

```python
class Blockchain:
    """동영상 파일을 나타내는 블록체인 클래스입니다."""

    def __init__(self):
        self.chain = [self.create_genesis_block()]

    def create_genesis_block(self):
        """제네시스 블록을 생성합니다."""
        return Block(0, b'', '0')

    def add_block(self, data):
        """새로운 블록을 추가합니다."""
        previous_block = self.chain[-1]
        index = previous_block.index + 1
        previous_hash = previous_block.hash
        block = Block(index, data, previous_hash)
        self.chain.append(block)
```