BankToken

6.24 김동은

계좌이체 자주 하시나요?

계좌이체도 많이 사용하지만 간편 송금을 주로 이용

출처 : Play Store

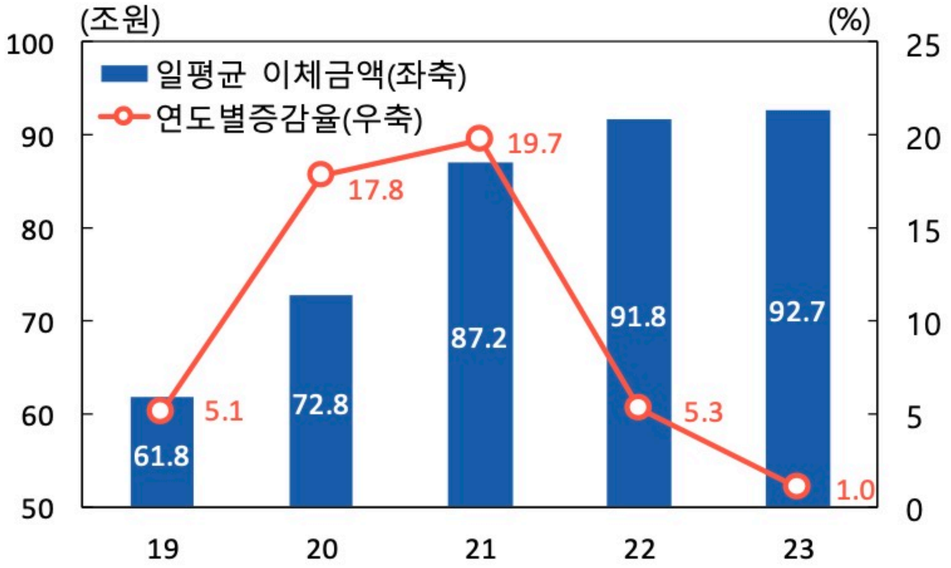
사실 계좌이체는 비싼 서비스

이용형태	단위/기준	일반고객	할인고객
창구 송금수수료 (같은 은행으로 보낼 때)	10만원 이하	면제	면제
	10만원 초과 ~ 100만원 이하	면제	면제
	100만원 초과	면제	면제
창구 송금수수료 (다른 은행으로 보낼 때)	10만원 이하	500원	400원
	10만원 초과 ~ 100만원 이하	2,000원	1,600원
	100만원 초과 ~ 500만원 이하	3,500원	2,800원
	500만원 초과	4,000원	3,200원
추심수수료	10만원 이하	2,000원	
	100만원 이하	3,500원	
	100만원 초과	5,000원	

출처 : KB국민은행

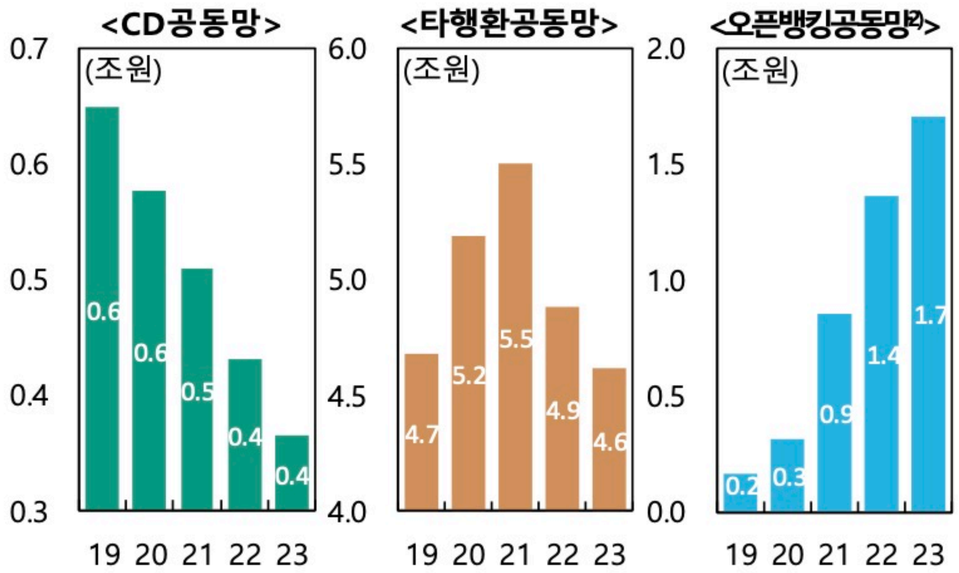
2023년중 소액결제망을 통한 계좌이체 규모는 일평균 92.7조원

연도별 계좌이체 금액 및 증감률[1)]



주: 1) 금액(일평균) 기준

여타 공동망별 이체금액[1)]

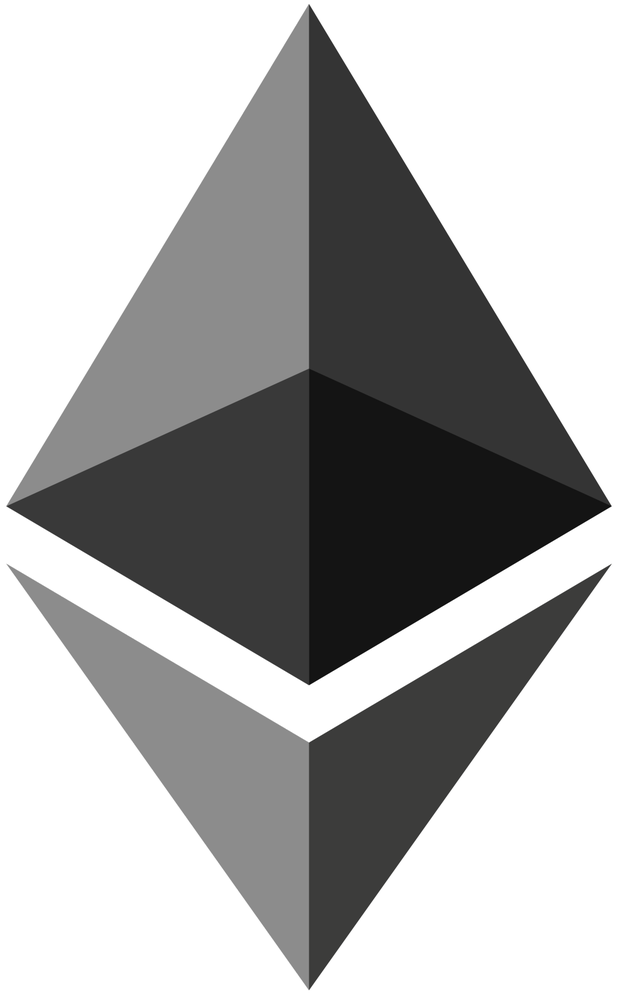


주: 1) 금액(일평균) 기준
2) 입·출금 각각 합계 기준

-> 하루 약 10억회의 계좌이체가 발생

출처 : 한국은행

계좌이체 비용을 줄여보자!
-> 전산망에서 인터넷으로 수단을 옮기자!



-> ETH 테스트넷으로 우선 구현

출처 : Crypto Logo

```
address public nodeMaintainer;
uint256 public constant NODE_REWARD = 10000 * 10 ** 8; // Adjusted to decimals (8)
uint256 public constant REWARD_INTERVAL = 1 days;
```

```
/* node와 bank 정보 mapping */
mapping(address => BankDeposit[]) public deposits;
mapping(address => Node) public nodes;
```

```
/* Role definition for node maintainer */
bytes32 public constant NODE_MAINTAINER_ROLE = keccak256("NODE_MAINTAINER_ROLE");
```

```
/* deposit func */
function deposit(uint256 amount, string memory bankId) public {    infinite gas
    require(amount > 0, "Amount must be greater than zero");

    // Log the deposit
    deposits[msg.sender].push(BankDeposit({
        amount: amount,
        bankId: bankId
    }));

    // Mint the corresponding amount of tokens to the depositor
    _mint(msg.sender, amount * 10 ** decimals());
}
```

```
/* withdraw func */
function withdraw(uint256 amount) public {    infinite gas
    require(amount > 0, "Amount must be greater than zero");
    uint256 amountInBaseUnit = amount * 10 ** decimals();
    require(balanceOf(msg.sender) >= amountInBaseUnit, "Insufficient token balance");

    // Burn the tokens
    _burn(msg.sender, amountInBaseUnit);

    // Calculate the amount to withdraw and log the withdrawal
    uint256 remainingAmount = amountInBaseUnit;
    uint256 withdrawalAmount;
    string memory bankId;

    while (remainingAmount > 0 && deposits[msg.sender].length > 0) {
        BankDeposit storage lastDeposit = deposits[msg.sender][deposits[msg.sender].length - 1];
        if (lastDeposit.amount <= remainingAmount) {
            withdrawalAmount = lastDeposit.amount;
            bankId = lastDeposit.bankId;
            remainingAmount -= lastDeposit.amount;
            deposits[msg.sender].pop(); // Remove the last element
        } else {
            withdrawalAmount = remainingAmount;
            lastDeposit.amount -= remainingAmount;
            remainingAmount = 0;
            bankId = lastDeposit.bankId;
        }
    }
```

```
/* Node reward function */
function re                                    ole(NODE_MAINTAINER_ROLE) {    infinite gas
    require                                    tRewardTime + REWARD_INTERVAL, "Reward interval has not passed");

    // Mint the reward tokens to the node
    _mint(node, NODE_REWARD);

    // Update the last reward time
    nodes[node].lastRewardTime = block.timestamp;
}

/* Block miner reward function */
function rewardBlockMiner() public {    infinite gas
    address miner = block.coinbase;
    require(miner != address(0), "Invalid miner address");

    // Define a fixed reward amount for the miner
    uint256 minerReward = 10 * 10 ** decimals(); // 블록 채굴시 10개의 보상을 지급

    // Mint the reward tokens to the block miner
    _mint(miner, minerReward);
}
```

contracts/BankTocken.sol 110:4

```
/* Transfer function with additional checks */
function transfer(address to, uint256 amount) public override returns (bool) {    infinite gas
    require(to != address(0), "Transfer to the zero address");
    require(amount > 0, "Transfer amount must be greater than zero");
    require(balanceOf(msg.sender) >= amount, "Insufficient balance for transfer");

    _transfer(msg.sender, to, amount);
    return true;
}

/* TransferFrom function with additional checks */
function transferFrom(address from, address to, uint256 amount) public override returns (bool) {    infinite gas
    require(from != address(0), "Transfer from the zero address");
    require(to != address(0), "Transfer to the zero address");
    require(amount > 0, "Transfer amount must be greater than zero");
    require(balanceOf(from) >= amount, "Insufficient balance for transfer");

    uint256 currentAllowance = allowance(from, msg.sender);
    require(currentAllowance >= amount, "Transfer amount exceeds allowance");

    _transfer(from, to, amount);
    _approve(from, msg.sender, currentAllowance - amount);
    return true;
}
```