# Security Survey of Bitcoin

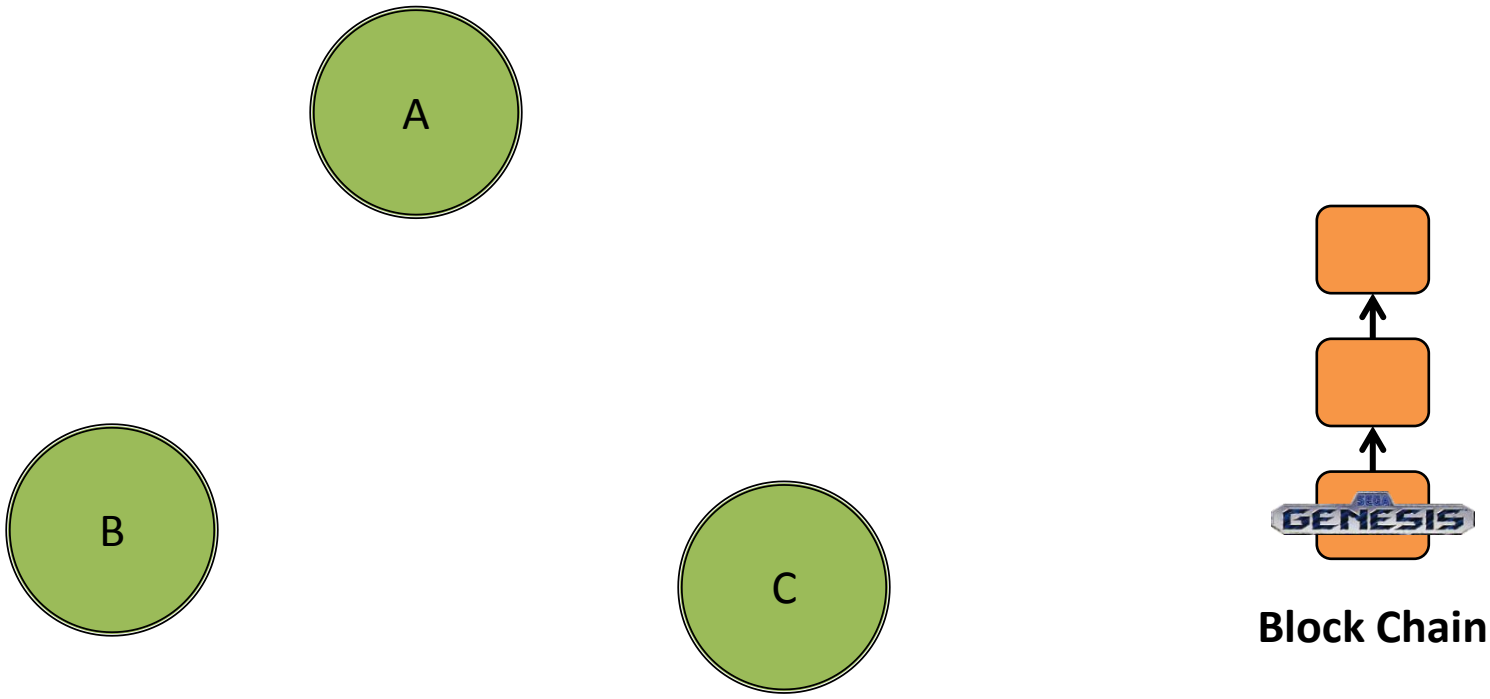
Ruven Chu and Andrew He

March 11, 2011

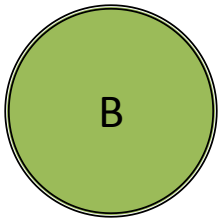
# Bitcoin background

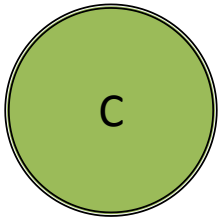
- Distributed digital currency, no centralized trust
- Coins are exchanged via broadcast messages (transactions)
- Transactions are gathered into blocks
- Nodes “approve” blocks by solving CPU-intensive problems
- Approved blocks are added to the block chain, which represents a complete timeline of transactions

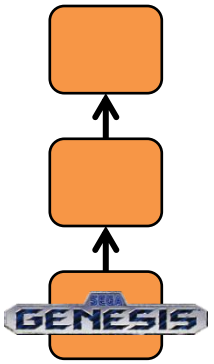
# A simplified example



# A simplified example

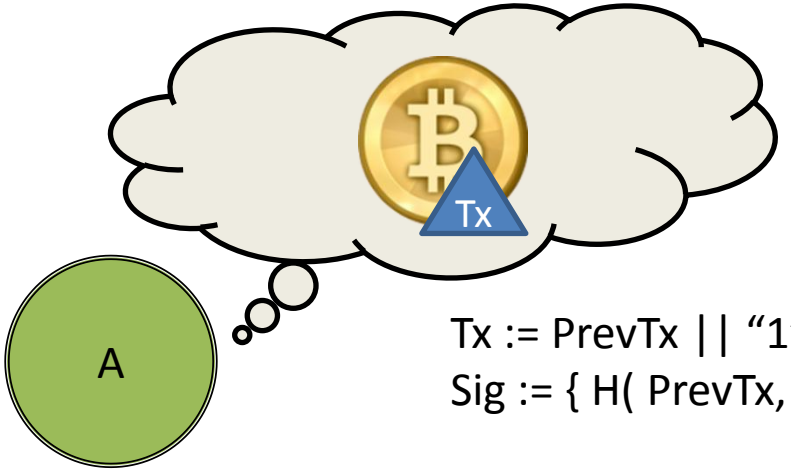



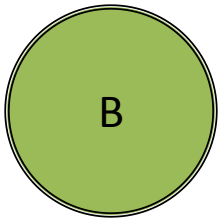

**Block Chain**

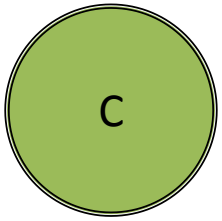
# A simplified example

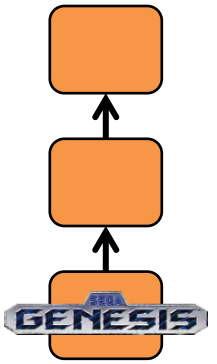



Tx := PrevTx || “1”, PK(B), Sig

Sig := { H( PrevTx, PK(B) ) }$_{SK(A)}$

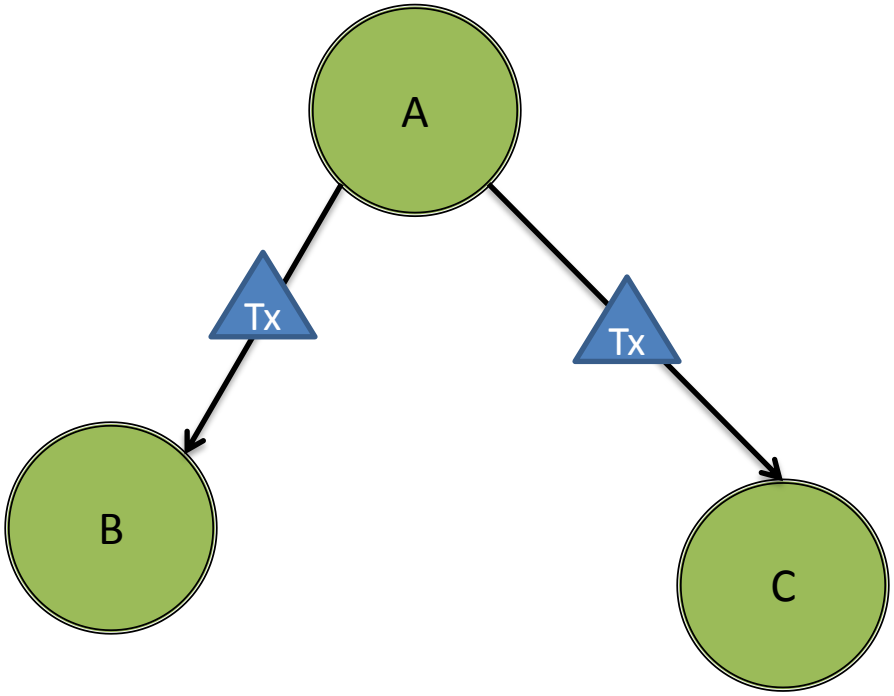


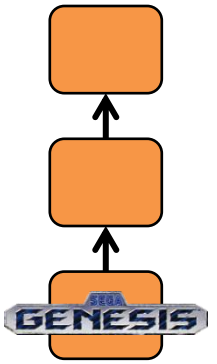




**Block Chain**

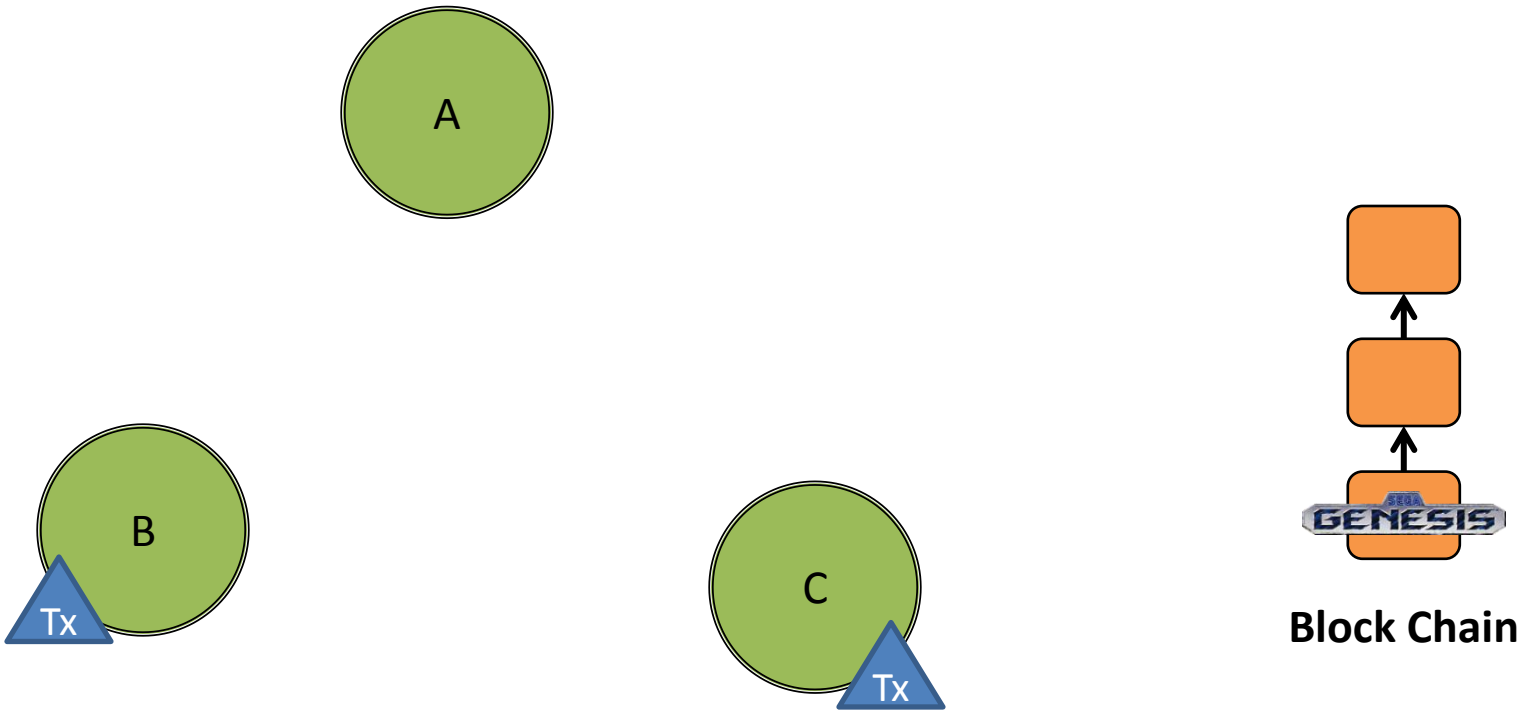
# A simplified example



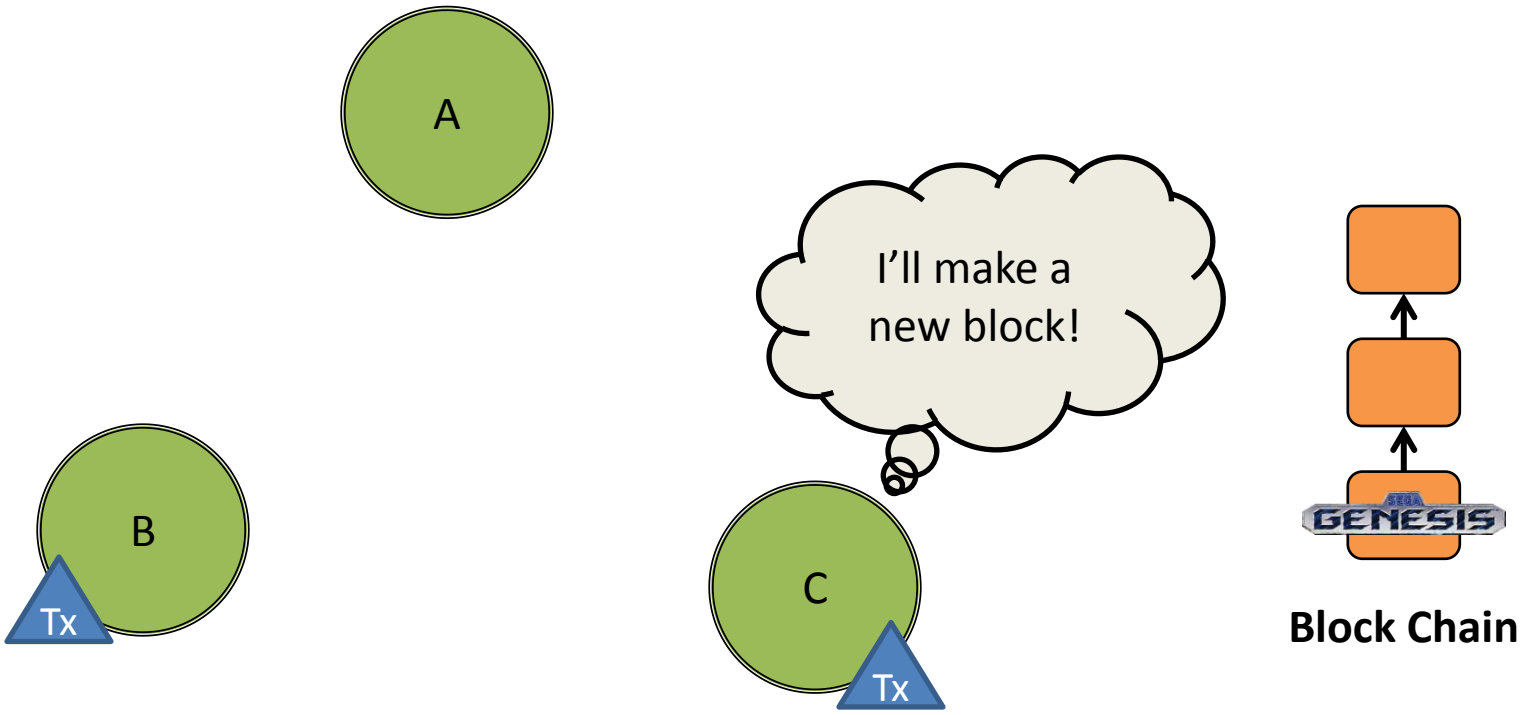




**Block Chain**

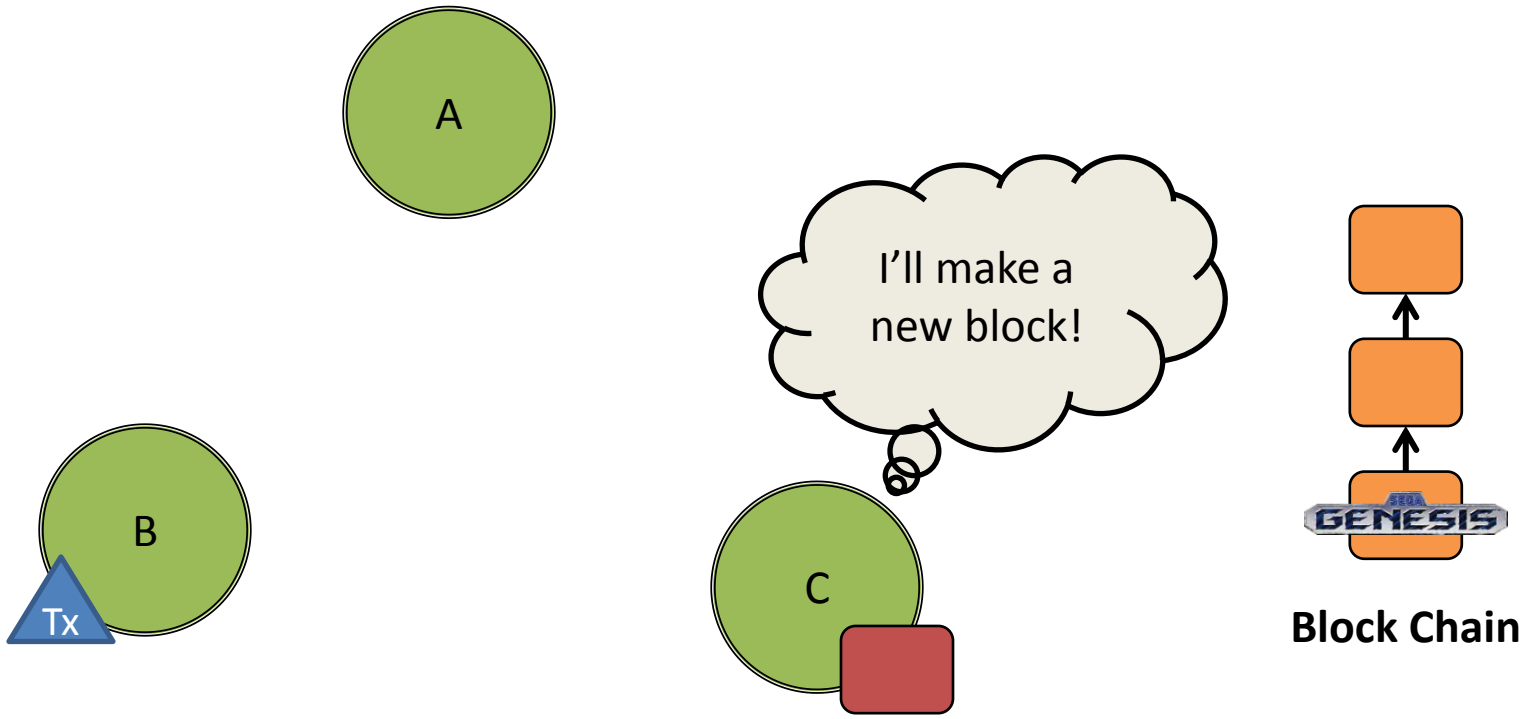
# A simplified example



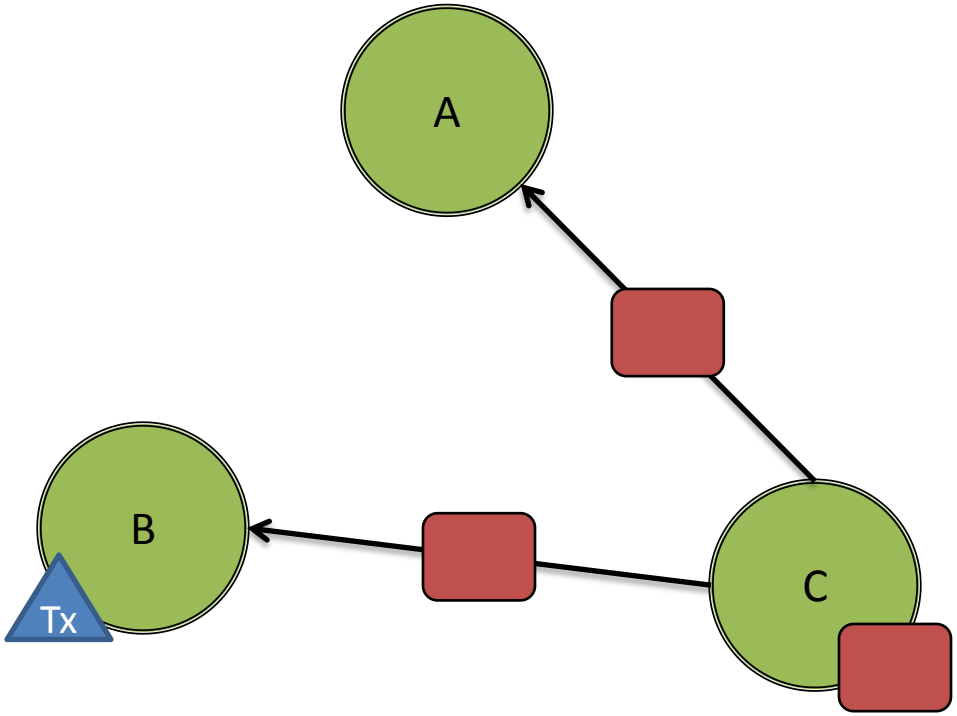
# A simplified example

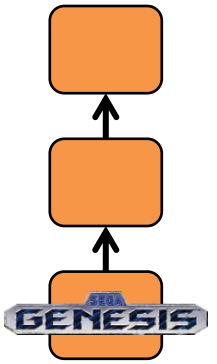


# A simplified example



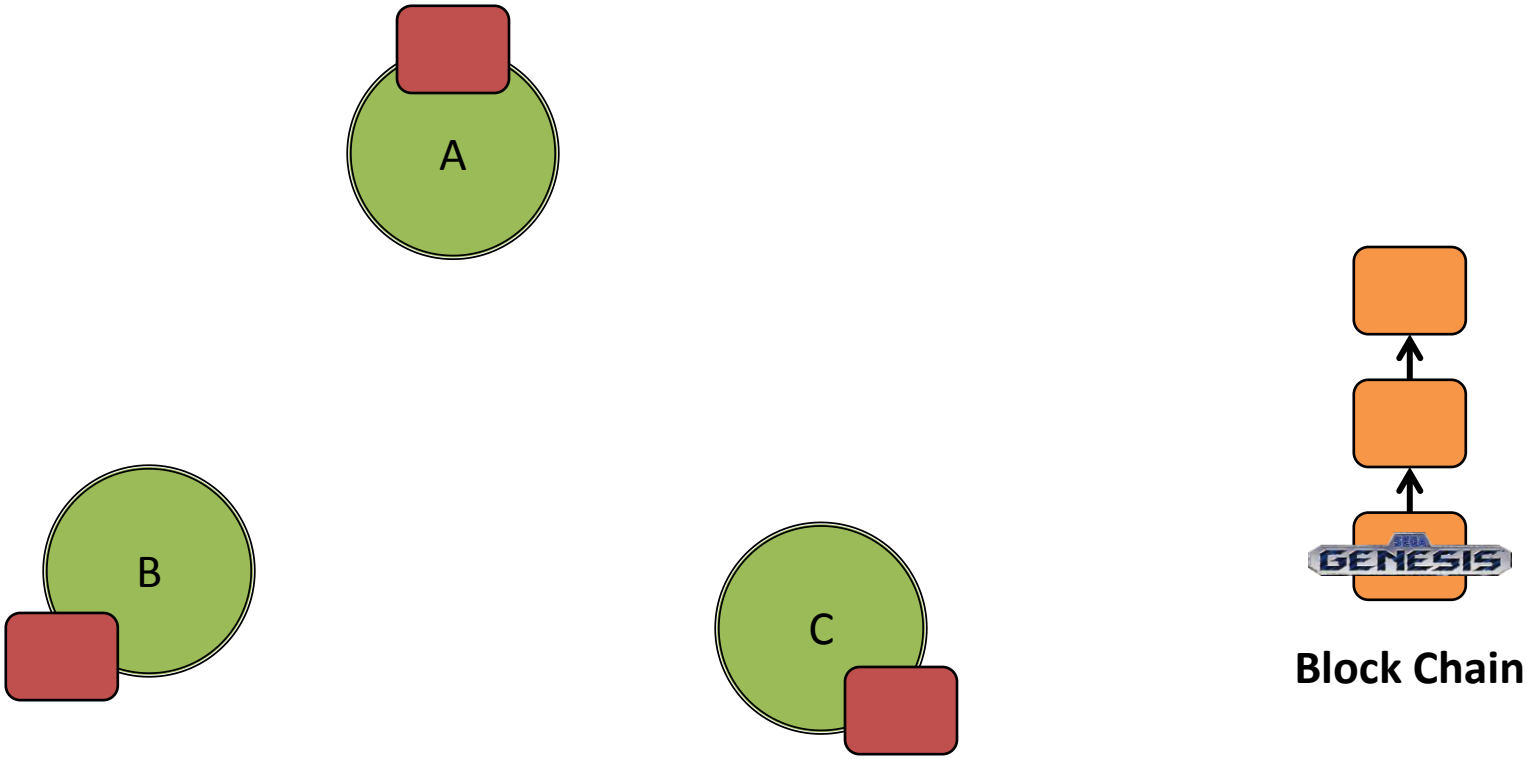
# A simplified example



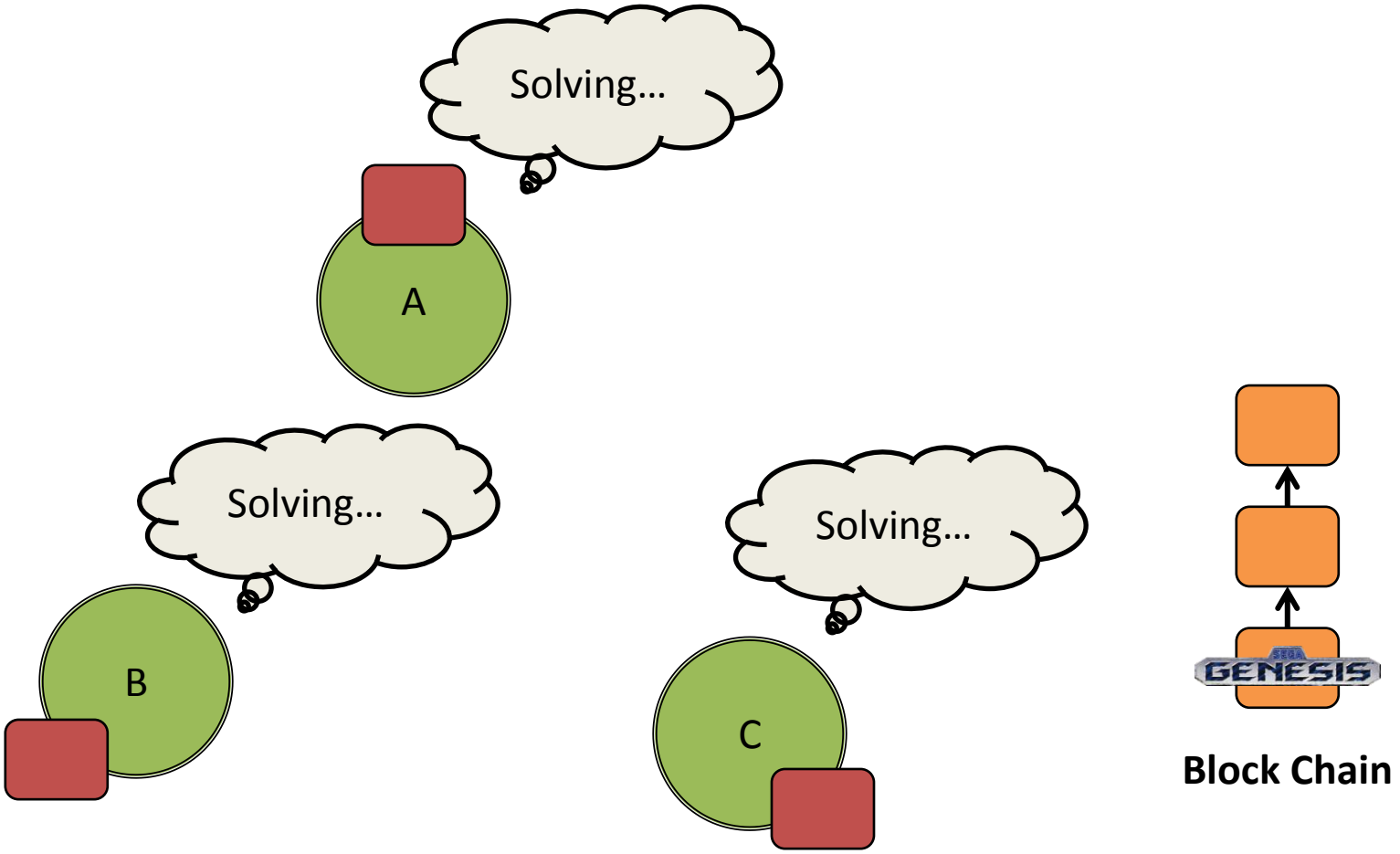




**Block Chain**

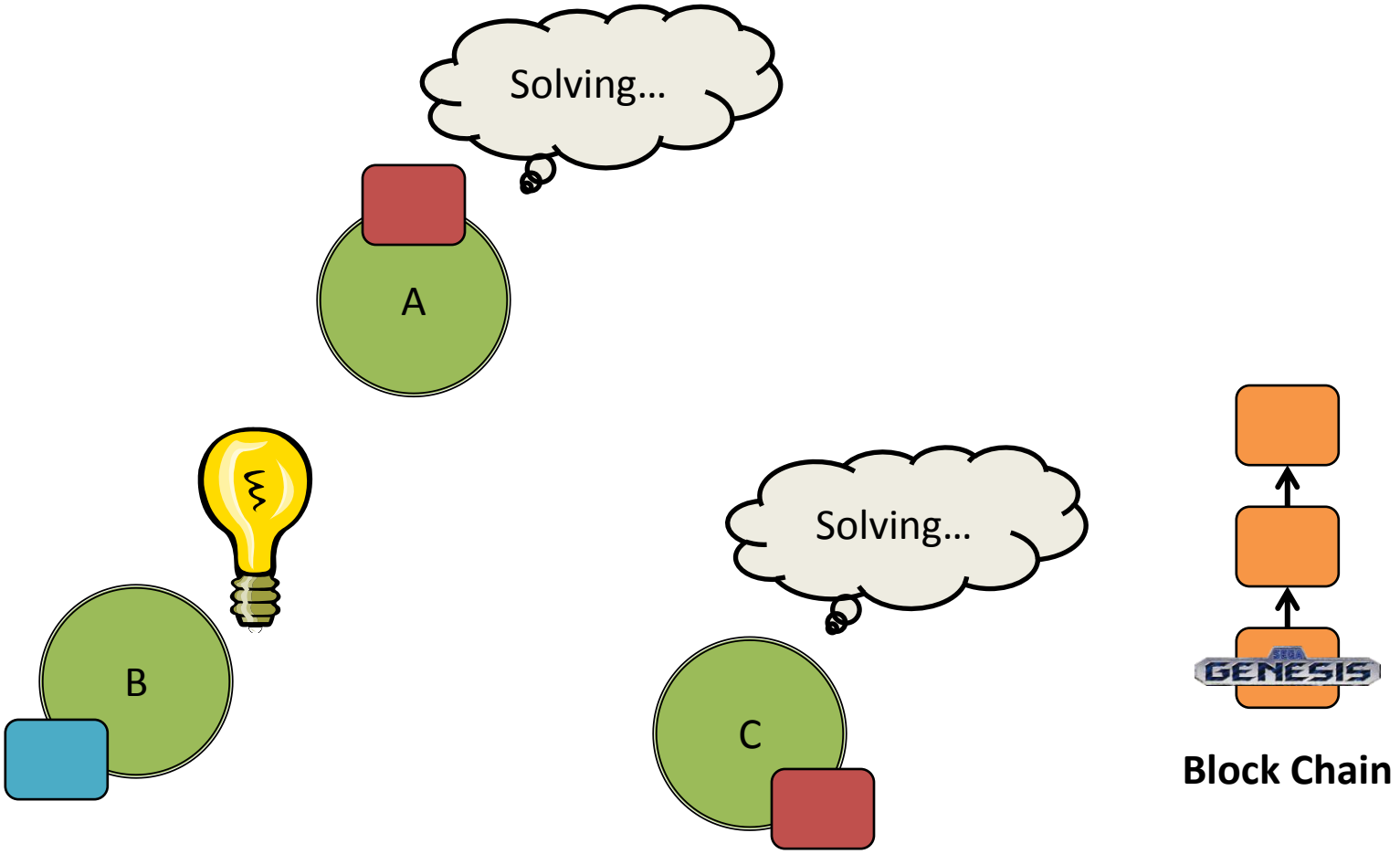
# A simplified example



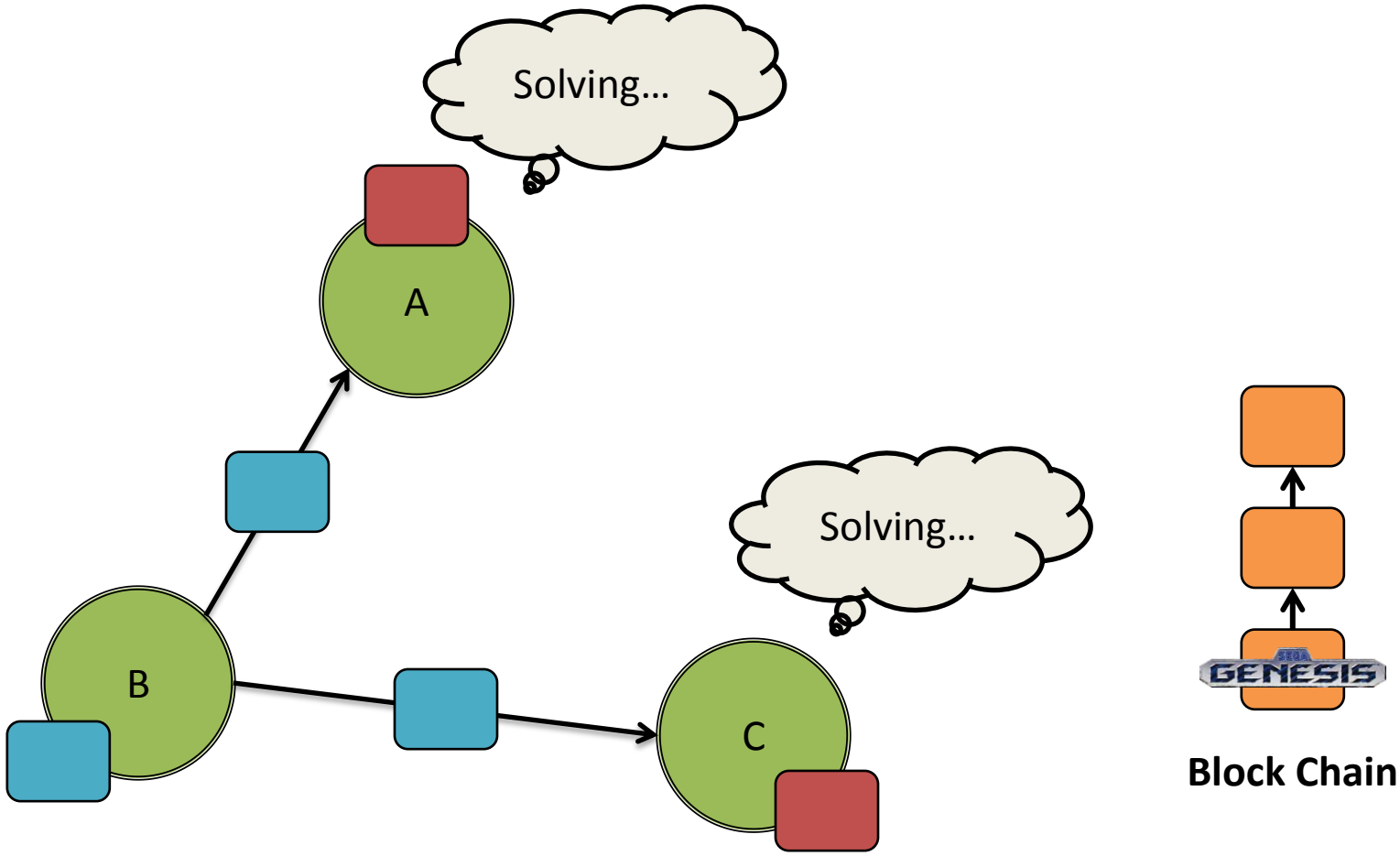
# A simplified example



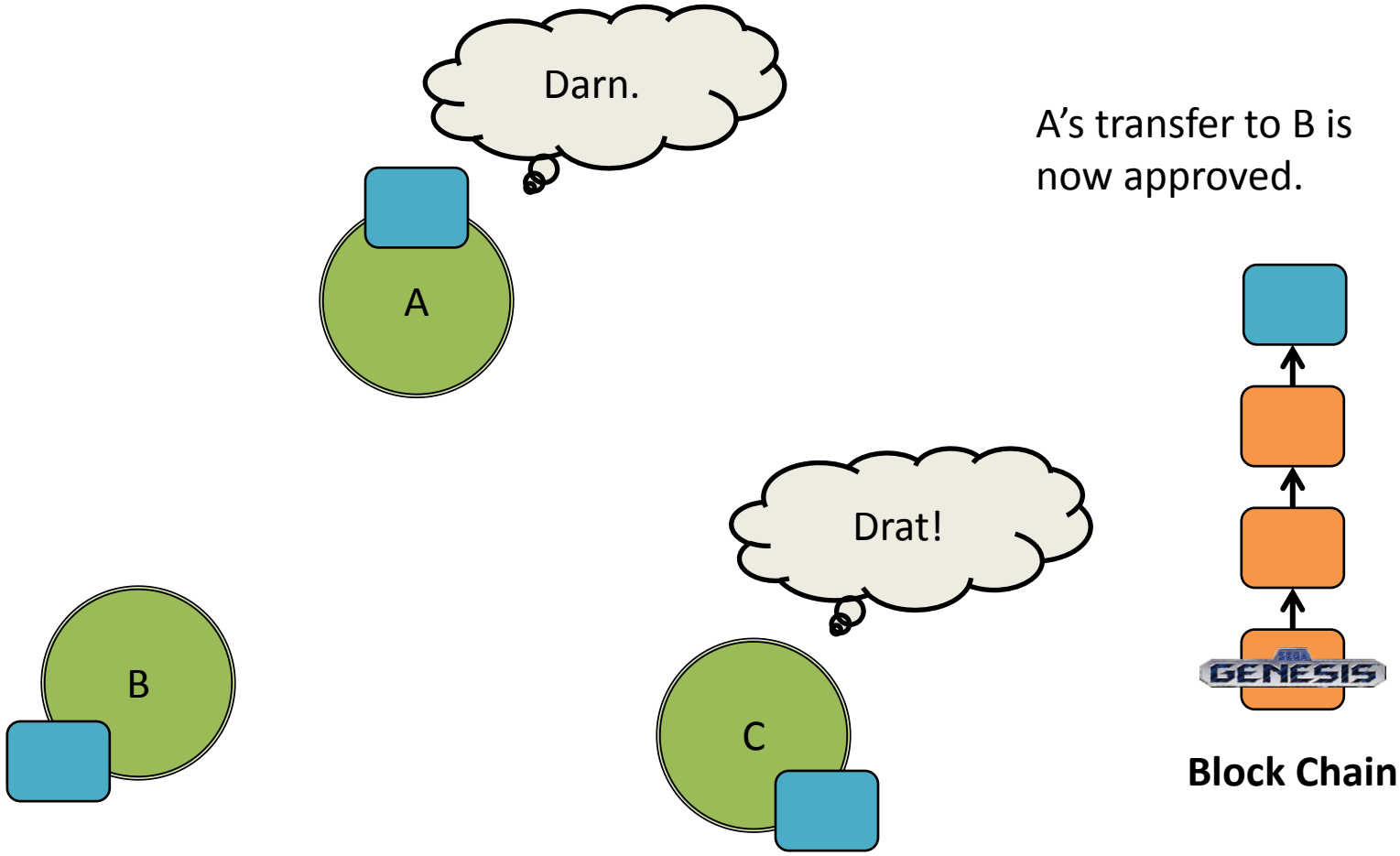
# A simplified example



# A simplified example

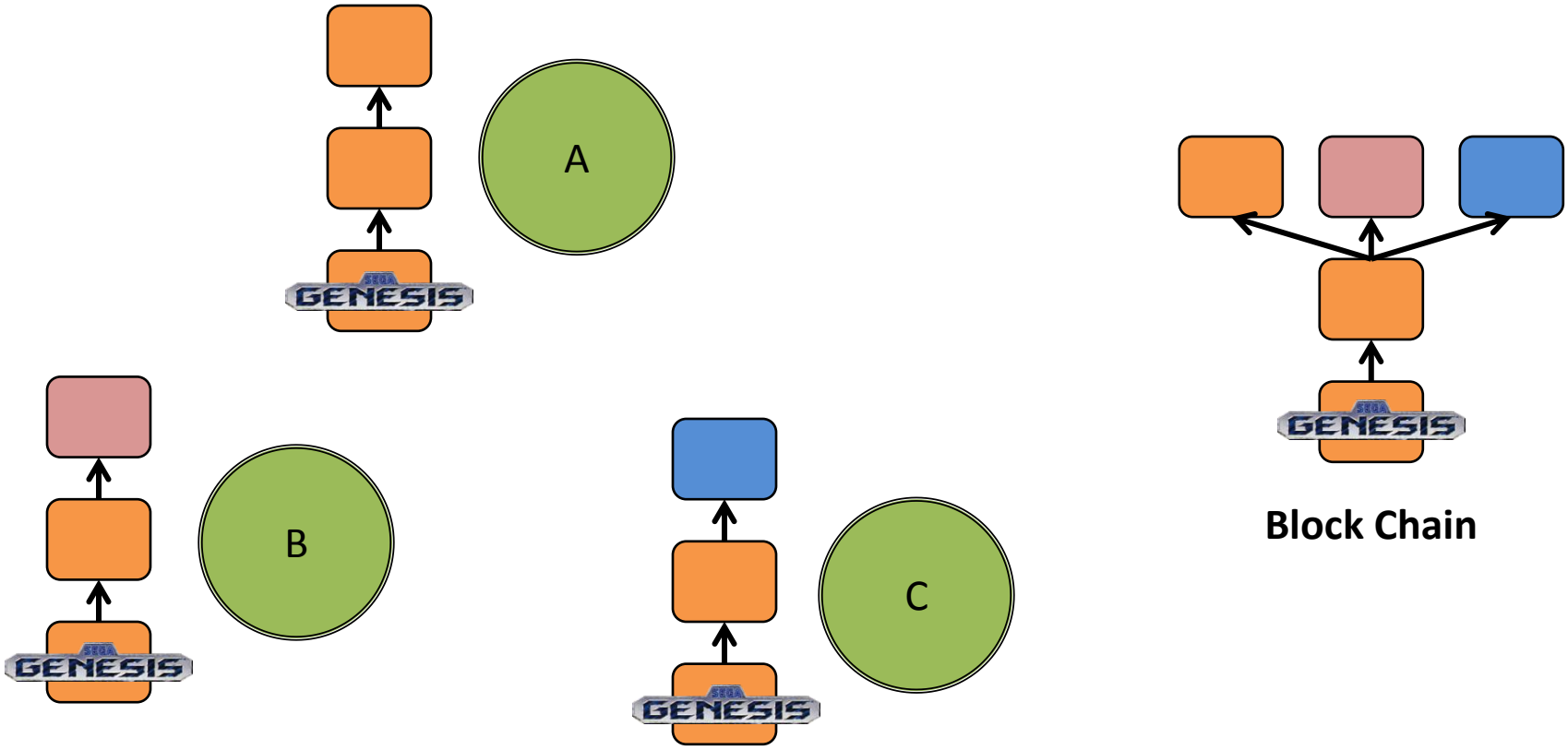


# A simplified example



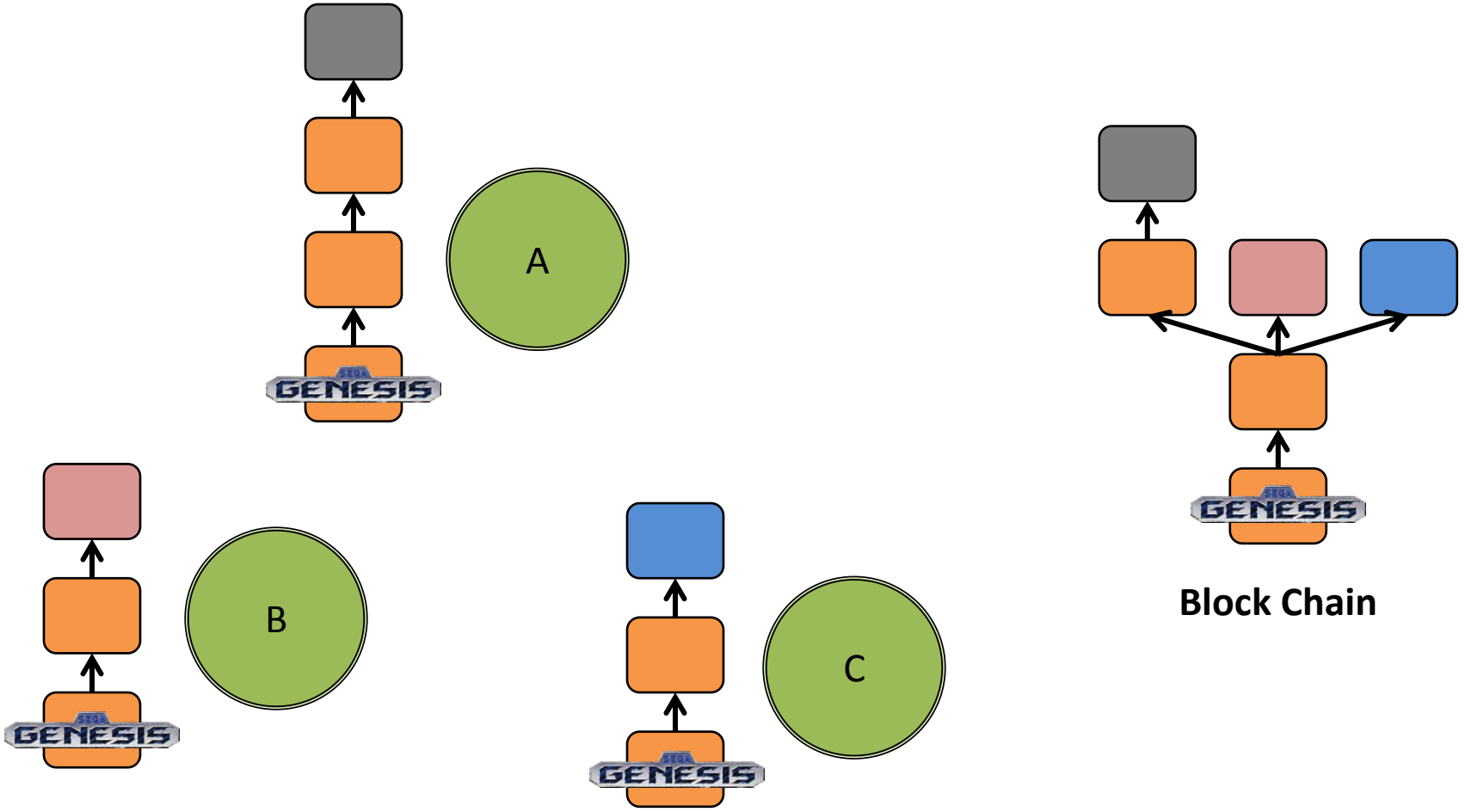
# Branching

- Nodes can actually have different chains from one another
- The block chain can branch to reflect these differences



# Branching

- Once one branch becomes longer, all nodes switch over



# The double-spending problem

- Node A might try to send the same coin to B and C
- Node A thus creates two conflicting transactions

# The double-spending problem

- Node A might try to send the same coin to B and C
- Node A thus creates two conflicting transactions

➢ Safeguard: a node that receives conflicting transactions will only accept the one transaction it sees first

# Security properties / threat model

- No branch of the block chain should include double-spends
- No node's "idea" of the block chain should include double-spends

- Model a network attacker:
  - This is realistic, since communications are unencrypted
  - Attacker is an "imperfect" broadcaster
- Include idea of CPU power in the model

# Murphi model

```
type
  TransId:          0..TransCount;
  ValidTransId:     1..TransCount-1;
  …
  CurrentTrans:     1..TransCount;
  WorkAmount:       0..MaxWork;
  AgentId:          union {ParticipatorId, IntruderId}
  BlockChain:       array[ValidTransId] of Message;
  TransChain:       multiset[TransCount] of ValidTransId;

  Agent : record
    chain: BlockChain;
    validchain: TransChain;
    …
    workleft: WorkAmount;
  end;

var
  age: array[AgentId] of Agent;
  cur: CurrentTrans;
  tra: BlockChain;
```

# Invariant

```
invariant "nobody legally double pays"
  forall i: ParticipatorId do
    forall j: ParticipatorId do
      i != j
      ->
      checknodoublepay(age[i],age[j])
    end
  end;
```

```
function checknodoublepay(one: Agent; two: Agent) : boolean;
var combined: TransChain;
var total: BlockChain;
var counts: CoinCounts;
begin
  undefine combined;
  undefine total;
  for i: ValidTransId do
    if MultisetCount(j:one.validchain,one.validchain[j] = i) > 0
| MultisetCount(j:two.validchain,two.validchain[j] = i) > 0  then
      MultisetAdd(i, combined);
      if !isundefined(one.chain[i].id) then
        total[i] := one.chain[i];
      end;
      if !isundefined(two.chain[i].id) then
        total[i] := two.chain[i];
      end;
    end;
  end;
  counts := getcounts(total, combined);
  for i:AgentId do
    if counts[i] < 0 then
      return false;
    end;
  end;
  return true;
end;
```

# Double-spending attacks

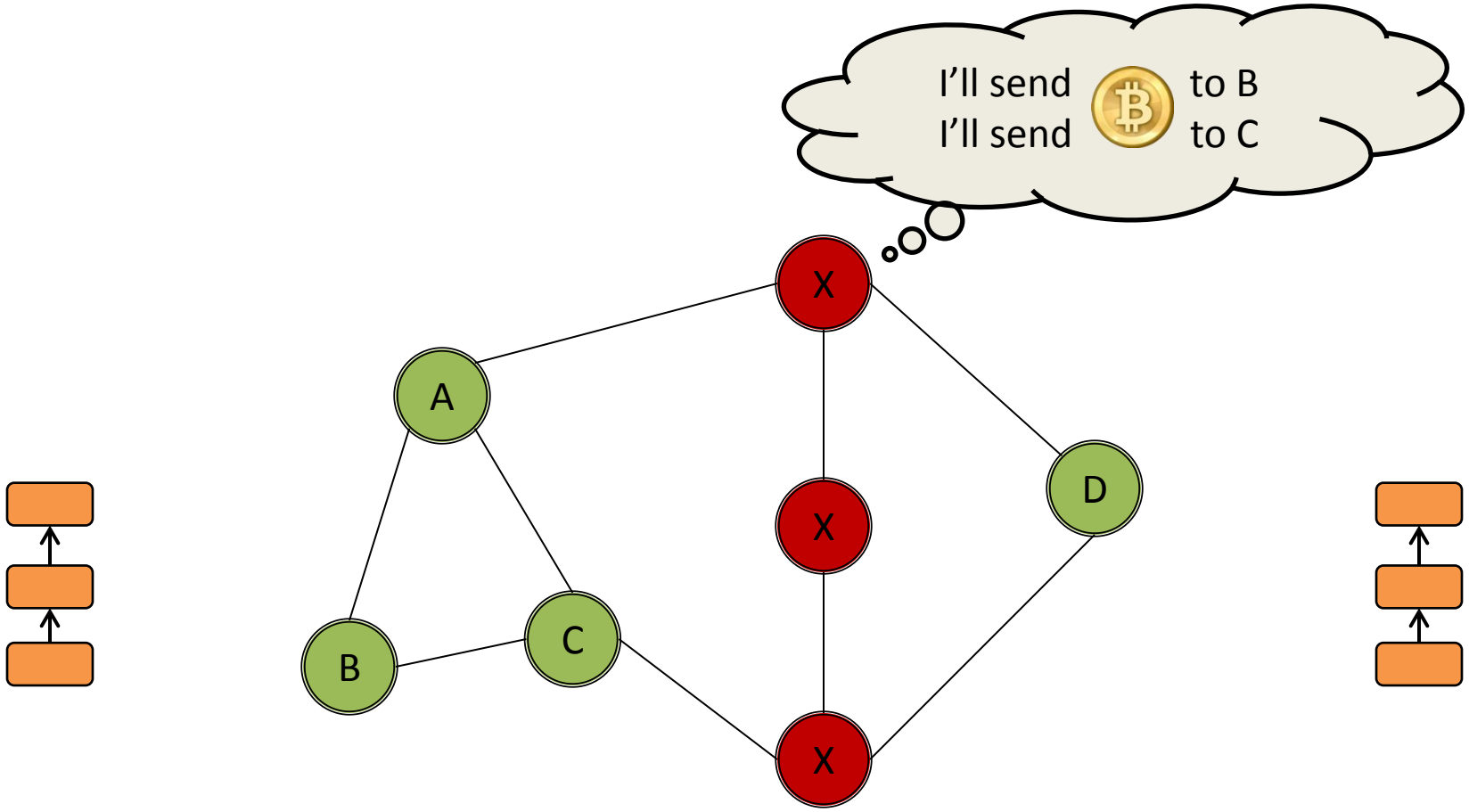
1) Network segmentation
2) Majority control of the network (w.r.t. CPU power)

- These are known attacks; our model didn't reveal any previously-unknown attacks.

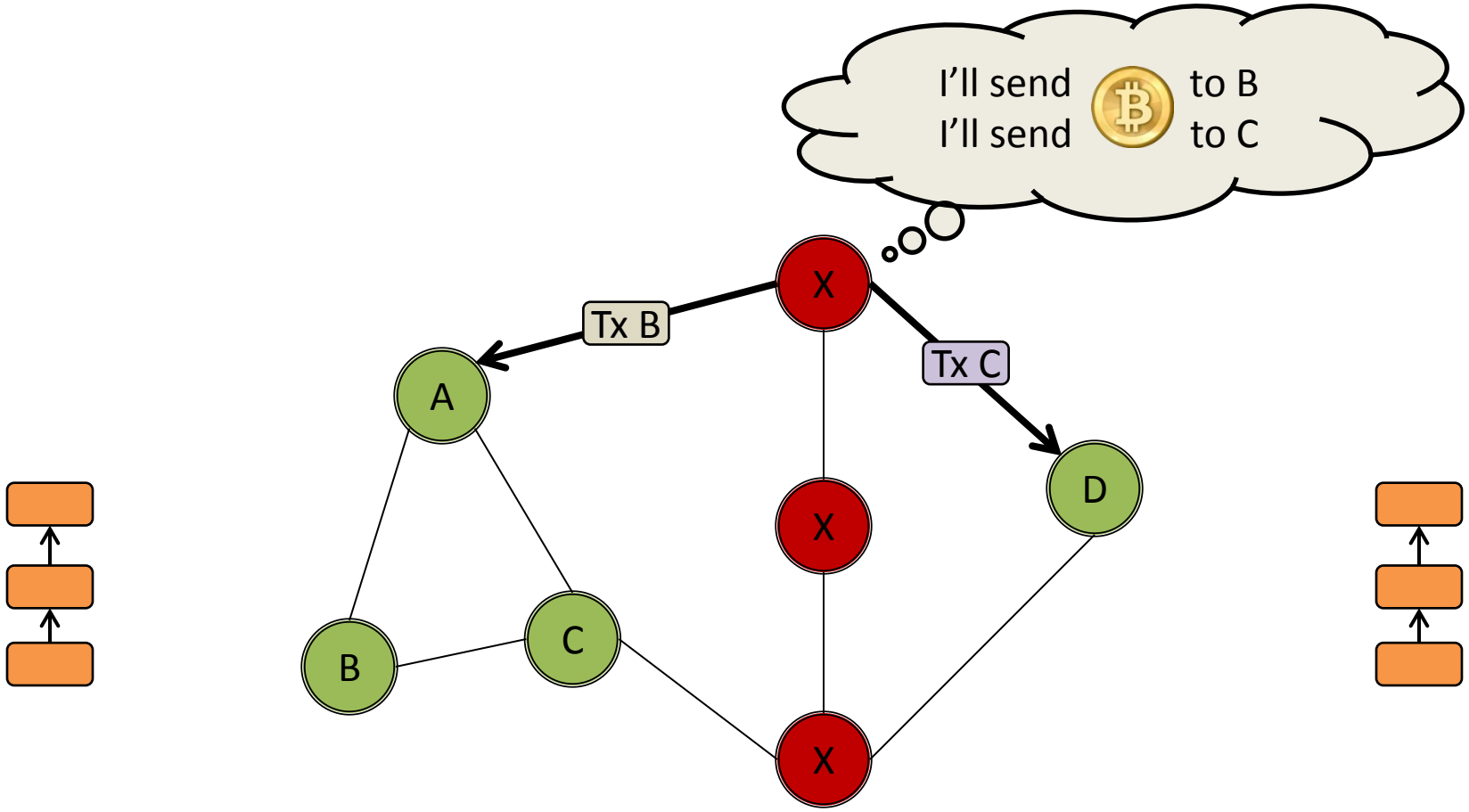
# Segmentation

- Attacker divides the network into two subgraphs that can’t communicate with each other
- Spend a coin twice: once in each subgraph

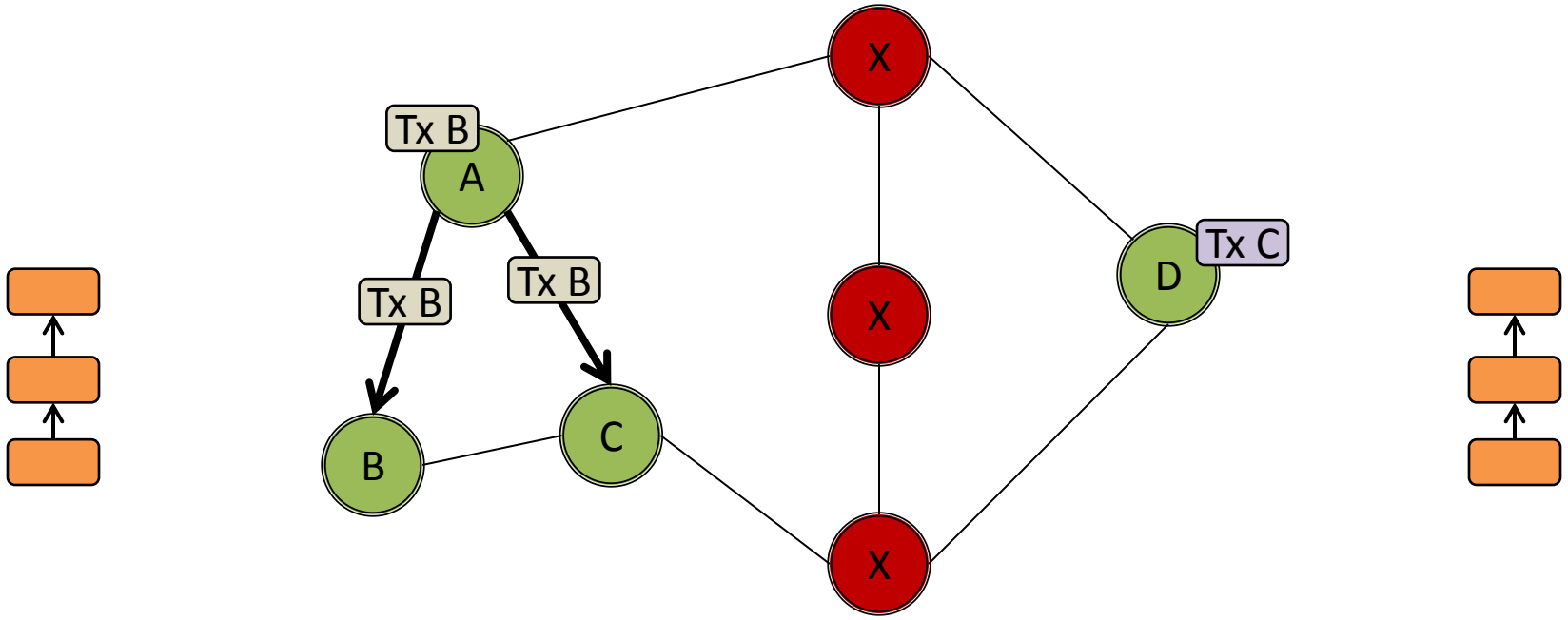
# Segmentation example



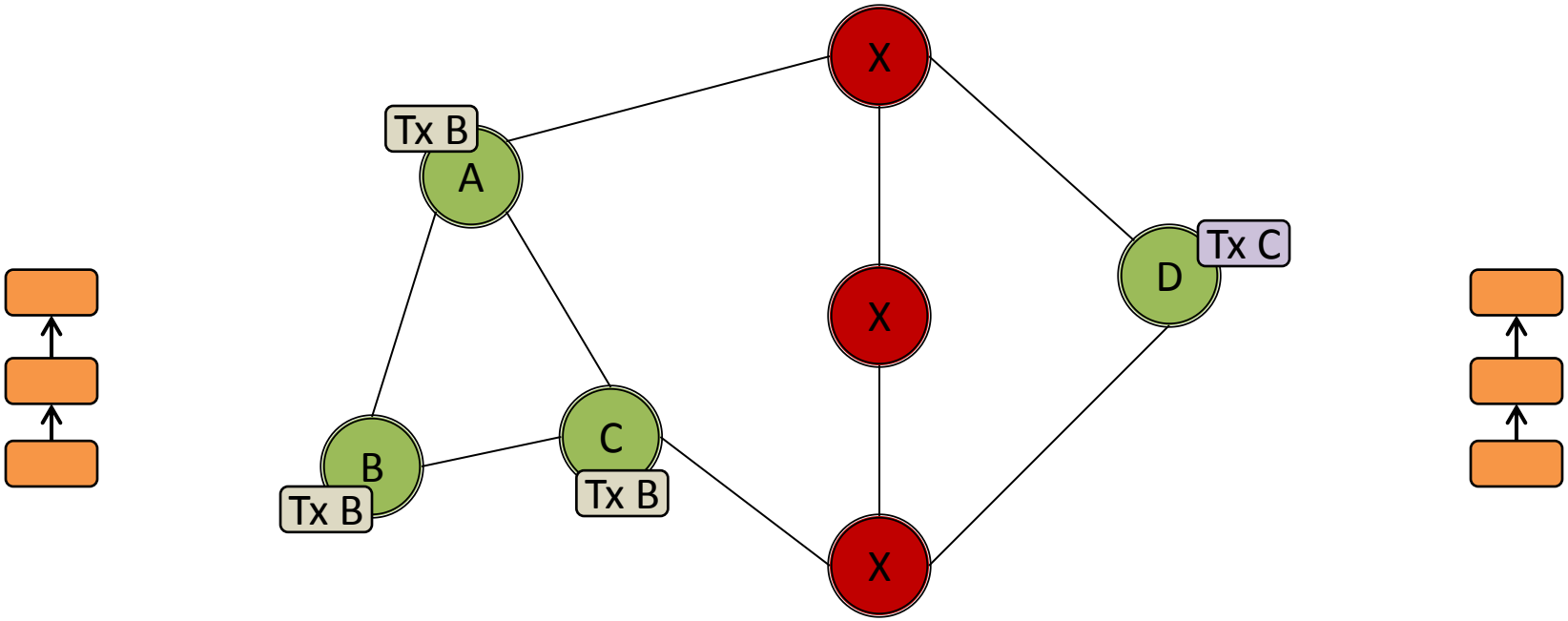
# Segmentation example



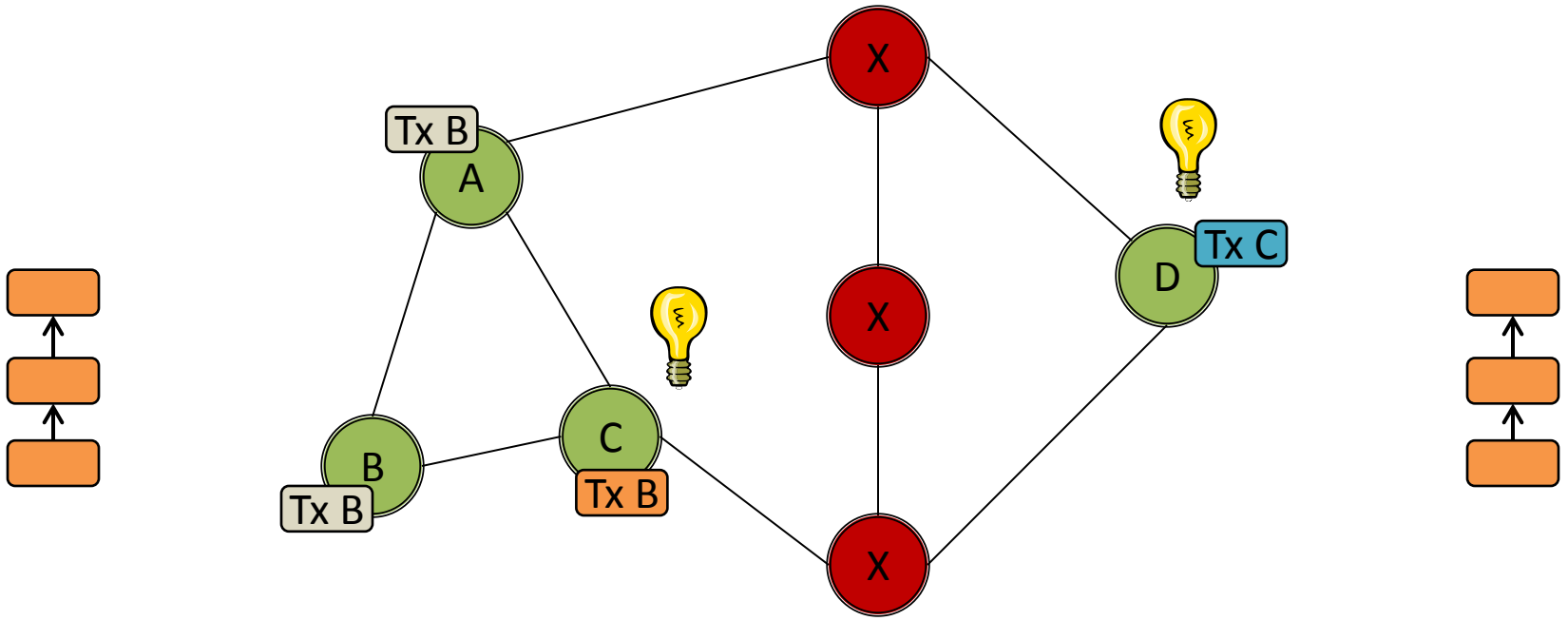
# Segmentation example



# Segmentation example

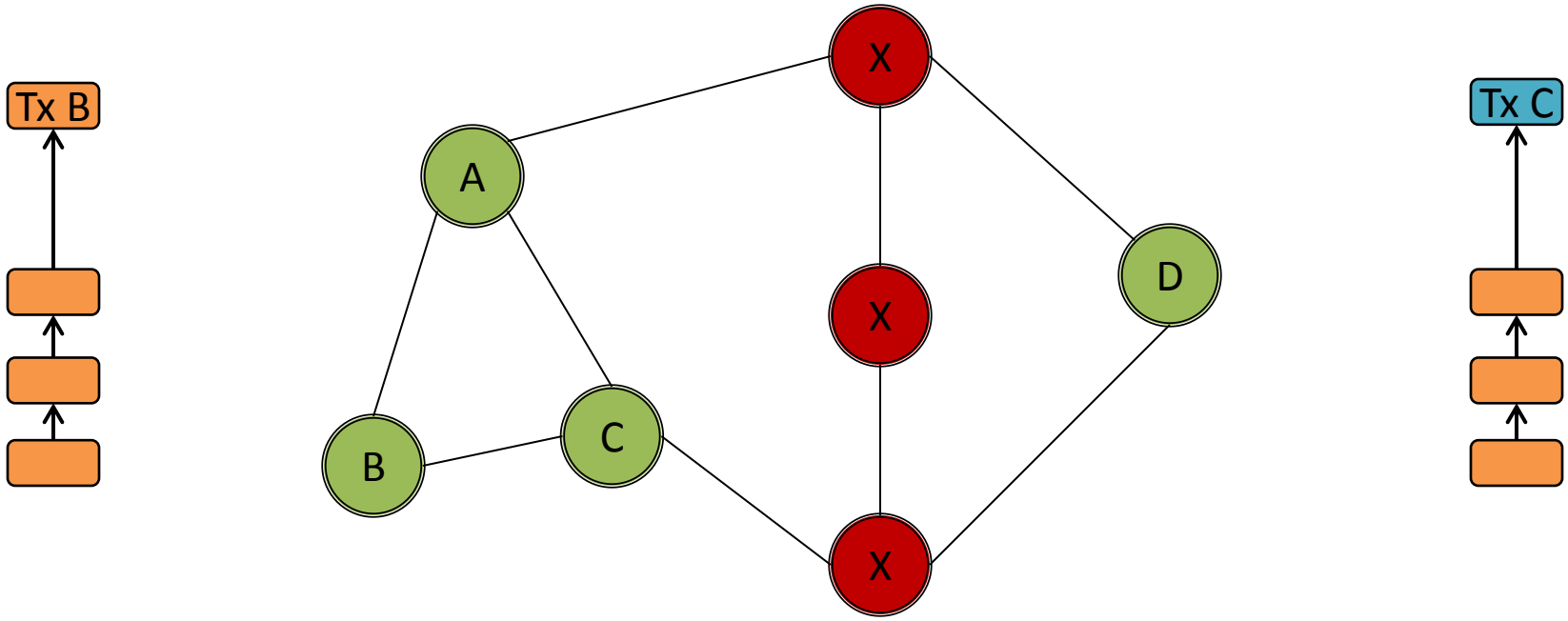


# Segmentation example



# Segmentation example

- After blocks propagate subgraphs, attacker has double-spent




➢ Actually feasible? Difficult, but not impossible to surround one or more nodes with attack nodes

# Majority power attack

- Attacker can grow whichever branch of the chain he wants, with whatever transactions he wants
- Trivially breaks Bitcoin for everyone

➢ Anyone with that much CPU is disincentivized to do this, since they could just gain from acting legitimately

# Other concerns

- Denial-of-service
- Anonymity

# Denial-of-service

- Follows from segmentation attack
- Trivial for a network attacker
- Bootstrapping process uses no encryption
  - Nodes connect to IRC channel to discover other nodes
- Attacker can connect you to dummy nodes that refuse to broadcast your messages

# Denial-of-service

- Follows from segmentation attack
- Trivial for a network attacker
- Bootstrapping process uses no encryption
  - Nodes connect to IRC channel to discover other nodes
- Attacker can connect you to dummy nodes that refuse to broadcast your messages

➢ Not considered a huge problem, just go somewhere else

➢ Attacker could cut off all of your other traffic too

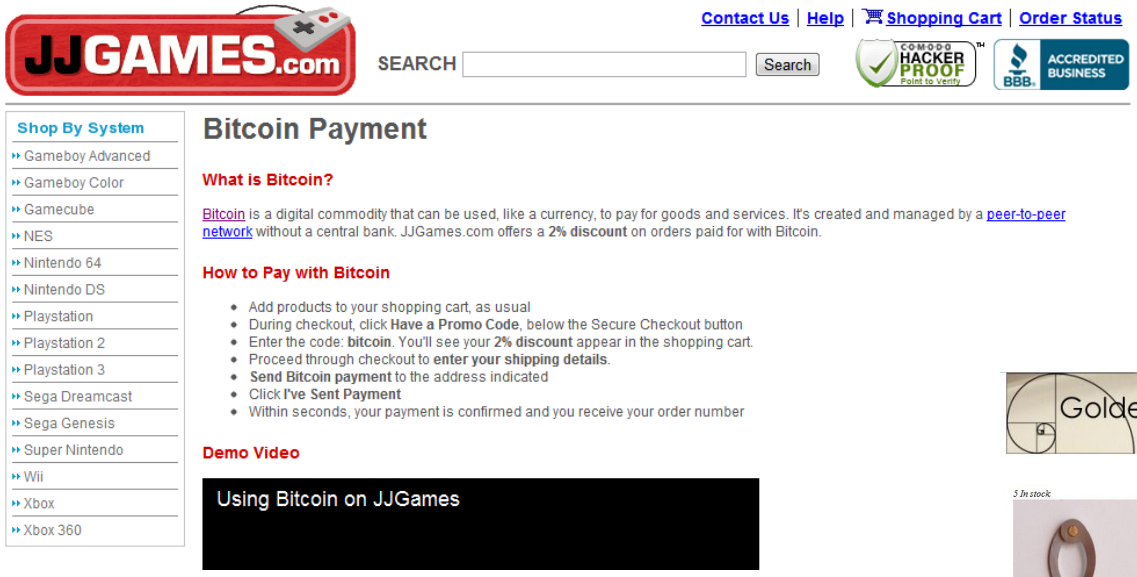
# Anonymity

- Transactions are anonymous only as long as your public key is never linked to any identifying information
- Impossible if you buy tangible goods shipped to your home

➢ Fix (somewhat in progress):

- Never send coins using the same public key twice
- Every time you receive coins, create a new identity (public key) and send your coins to that new identity through a “mixer”
- The “mixer” accepts lots of coins from various people and randomly matches source(s) to destination(s)

# Bitcoin adoption





# Bitcoin adoption