Intro

1) How a blockchain works (plumbing, not poetry)

Think of a blockchain as a **global append-only database** that many strangers keep in sync without a central admin. Here's the life of a transaction and the mechanisms that keep it honest.

A. What a transaction is

- A transaction is a small message: "Alice authorizes 1 coin to Bob."
- Alice proves it's really her by attaching a **digital signature** (made with her private key). Anyone can verify it with her **public key**.
- On different chains the accounting model differs:
  - **UTXO model** (Bitcoin): you spend specific "coins/outputs" you received earlier.
  - **Account model** (Ethereum): each account has a balance; debits/credits change it.

B. How transactions enter the system

- Alice's wallet broadcasts the signed transaction to the peer-to-peer network.
- Nodes do **cheap checks**:
  - Signature valid?
  - Inputs unspent (UTXO) / balance sufficient (account model)?
  - Nonce correct (prevents replay), fees present?
- Valid ones sit in a public waiting room called the **mempool**.

C. Blocks: batching and ordering

- A **block** = header (metadata + hash of previous block + Merkle root of txs) + list of transactions.
- The block links to the previous block via a **hash** → that's the "chain."

D. Who gets to add the next block? (Consensus)

Different blockchains make writing to the ledger **costly** (to stop spam/cheating) but keep reading/verification **cheap**:

- **Proof-of-Work (PoW, e.g., Bitcoin):**
  Many miners race to find a lucky **nonce** so the block header's hash is below a target. Finding it costs **electricity** (millions/billions of hash trials). Verifying the winner is **trivial** (one hash and compare). This "asymmetric cost" is why it's open yet robust.
- **Proof-of-Stake (PoS, e.g., Ethereum today):**
  Validators lock up (**stake**) the native coin. One proposes a block; others **attest**. If they misbehave, their stake can be **slashed**. Cost here is **economic collateral**, not electricity. Verification is still cheap (check signatures/attestations).
- **BFT-style consensus (e.g., Ripple/XRP-style or some permissioned chains):**
  A known set of validators vote and need a **supermajority** to finalize a block. Fast finality, but validator membership is curated.

E. Forks, reorgs, and "finality"

- Sometimes **two blocks** at the same height appear (e.g., two miners win near-simultaneously). The chain "forks" briefly.
- Nodes follow the rule "**longest/most-work (or most-weight) chain wins**." As soon as one branch gets another block, the network converges on it.
- If your transaction was in the loser branch, it gets **reorganized** out (a **reorg**). It should be included again later, but that brief window is why we wait for **confirmations**.
- **Probabilistic finality (Bitcoin):** Every block after yours drops the chance of reorg exponentially. The common heuristic "**6 confirmations**" ≈ ~1 hour makes it practically irreversible.
- **Deterministic finality (many PoS/BFT systems):** Once enough validators sign a checkpoint (supermajority), it's **final** in seconds; no reorgs beyond that point unless the validator set itself is slashed/rolled back.

F. "Immutability," precisely

- Each block's header contains the **hash of the previous block**. Change *anything* in an old block → its hash changes → every later block's hash changes → the chain breaks.

- To tamper with history, an attacker must **rewrite that block and all after it** and outpace honest participants (i.e., >50% hashpower in PoW or control a supermajority in PoS/BFT). That **economic infeasibility** is what we mean by "immutable."

Applications

1) fast payments across countries (remittances)

the problem with today (banks):

- banks in different countries don't share one notebook. so they rely on **messages** ("please pay this person") that hop through several banks. messages ≠ money.
- to make payouts fast, big banks keep **pots of foreign cash** with each other. this is the scary word pair **nostro/vostro**:
  - picture bank-A in india keeping a **jar of pesos** sitting inside bank-B in mexico. that jar is "nostro" for A (our money with you) and "vostro" for B (your money with us).
  - jars must be filled **before** payments happen. that ties up tons of money doing nothing.
- delays happen because: time zones, multiple banks in the chain, compliance checks at each hop, and sometimes manual reconciliation. days, not seconds.

how blockchains help:

- you swap rupees → a crypto token on an exchange in india, **send the token on-chain in seconds**, then swap token → pesos on an exchange in mexico, and pay out locally.
- no pre-filled jars. money is **just-in-time**: buy → send → sell.
- the blockchain gives you the shared, final "yes, this transfer happened" in seconds. fewer hops, less trapped cash.

who earns/pays? sender pays tiny network fees and a small FX spread; exchanges earn a spread; the blockchain's referees earn a fee for stamping the page.

2) swapping one token for another (decentralized exchange = DEX)

imagine a fruit stand that always keeps apples and bananas. there's a signboard with a simple rule:

> "price adjusts automatically so the apple×banana inventory stays balanced."

on-chain:

- two piles (liquidity pools) live inside a **smart contract** (think: tamper-proof vending machine). you put in apples, it gives you bananas; the machine updates its signboard price **by formula**.
- the machine takes a tiny fee per trade, which is paid to the people who stocked the piles (liquidity providers).

why blockchain? no shopkeeper needed. rules are public, moves settle in seconds, anyone can provide stock and earn fees.

who earns/pays? traders pay a small fee; liquidity providers earn those fees; the chain's referees earn gas (execution) fees.

3) borrowing & lending without a bank

think of a **robotic pawn shop**:

- you deposit gold worth ₹120 (your **collateral**) into a smart-contract "safe."
- the safe automatically lets you **borrow ₹60** of a stable digital rupee.
- if your gold price falls too much, **the robot sells just enough gold** to cover your loan. no arguments, no phone calls—pure rules.

why blockchain? the safe is a public program everyone can inspect; rules can't be bent; the money moves automatically when conditions hit. this cuts human back-office work and time.

who earns/pays? borrowers pay interest; lenders earn interest; the protocol may also give reward points/tokens; the chain's referees earn gas fees.

4) tickets, memberships, and digital goods (NFTs)

picture a concert ticket as a unique digital stamp:

- it sits in your wallet; the venue's door scanner checks the **blockchain** to see who owns stamp #123.
- you can resell it in a marketplace; the transfer is instant, and if the artist wants, **a small resale royalty** is paid automatically back to them (because the rule is inside the ticket's smart contract).

why blockchain? proof of ownership that's easy to verify, hard to fake, and simple to transfer. creators can bake in royalties. marketplaces earn only for matching buyers/sellers, not for keeping big private databases.

who earns/pays? buyer/seller pay marketplace fees; creators can earn automatic royalties; referees earn network fees.

5) supply chain tracking (where did my mango come from?)

every time the mango changes hands (farm → packer → ship → warehouse → store), someone writes a short **event** into the shared notebook:

- "batch A picked at farm X at 7:05am."
- "sealed box #445 at 11:20am."
- "temperature 4.0°C during shipping."

now, when there's a quality problem, no chasing 12 companies' databases—**one timeline** for everyone. you choose whether that chain is public or only for approved companies (a "permissioned" blockchain).

who earns/pays? companies pay for the software layer that makes writing/reading easy; value shows up as fewer recalls, less fraud, faster audits.

the two ideas that unlock all this

1) smart contracts = vending machines that hold money

- a smart contract is just code sitting at a public address. it can hold assets and has rules like "if X happens, pay Y."
- nobody can change the rules after it's deployed (unless the rules *themselves* say upgrades are allowed).
- **ethereum** is the biggest city of these vending machines. every action costs a tiny fee called **gas** (paid in **ether/ETH**)—like putting a coin in a real vending machine so it can whirr its gears.

why ethereum feels unique:

- huge library of prebuilt parts (token standards, wallets, tools) → teams ship fast.
- lots of other blockchains copy the same "engine" to be compatible because developers already know it.
- fees burn a little ETH each time, and thousands of independent referees keep the city honest.

2) finality = "the ink has dried"

- the moment something lands on the latest page, it's *probably* fine but could be changed if two pages were written at the same time and the network picks the other one.
- after a few more pages are glued on top, the ink is considered **dry**. some blockchains dry in ~1 hour, others in ~5–15 seconds. apps choose how many pages they wait based on risk.

① Understanding the Money Market

1) What is "fiat" money?

Think of **fiat** as "national money by government decision." It's the rupee, dollar, euro—paper or digital

Think of **fiat** as "national money by government decision." It's the rupee, dollar, euro—paper or digital balances created by a central bank and accepted because the state says "this settles debts," taxes are paid in it, and everyone else accepts it too. There's no gold bar behind each note; value rests on trust in the issuer and its rules. Investopedia

Why it matters: prices, salaries, and loans are denominated in fiat; governments can create more/less of it, which links directly to **inflation** and interest rates. (More money chasing the same goods tends to push prices up; raising interest rates can cool demand.)

2) How do banks move money across borders? What is SWIFT?

Picture SWIFT as a **global postal service for payment instructions**. It **doesn't move money**; it moves standardized **messages** between banks: "Debit my customer, credit yours." Actual funds settle through the banks' accounts with each other or with central banks. SWIFT is a member-owned co-op that provides the secure messaging network most banks use. Swift +1

The "jars on each other's shelves" (nostro/vostro) — plain-English

If an Indian bank often pays out pesos in Mexico, it will keep a **jar of pesos** parked at a partner bank in Mexico (and vice-versa). That's the old system's way to make payouts **fast at the destination**—but the price is **trapped working capital** in many countries and multiple hops of checks/messages. Each hop adds time-zone waits, compliance checks, and fees. (No citation needed for the analogy; it's a simple explanation of the well-known correspondent model.)

3) What is a currency "peg" in fiat — and how is that different from a crypto "peg" (stablecoin)?

Fiat currency peg (government-run): a country promises its money trades at a fixed rate to another (e.g., Hong Kong keeps HKD ≈ 7.75–7.85 per USD). The **central bank defends** this by buying/selling its currency using large **foreign-exchange reserves**. That policy choice constrains interest rates and money creation. Hong Kong Monetary Authority IMF eLibrary

Crypto stablecoin "peg" (issuer-run): a company issues digital tokens redeemable 1:1 for dollars in its bank accounts (e.g., **USDC**). If the issuer really holds safe cash/T-bill reserves and lets you redeem, the token tends to trade at $1; the **redemption window** is the anchor. This is **not** a government promise; it's an issuer promise with audits. (Algorithmic "pegs" without solid backing, like **TerraUSD** in 2022, can fail spectacularly.) Circle +1 Reuters

Why pegs matter: a **fiat peg** is monetary policy (state defends a rate); a **stablecoin peg** is a product design (issuer defends redemptions). They look similar to users ("$1"), but the machinery and risks are different.

4) Inflation & governance — who decides the rules?

- **In fiat:** the **central bank** sets interest rates and money supply targets; the elected government sets tax/spend. That's how inflation and growth are managed (or mismanaged).
- **In public crypto (preview):** rules are mostly **in code** (fixed or formula-based issuance, fee rules, validator rewards). For example, Ethereum's fee design **burns a portion of fees**, so supply can shrink when activity is high; that's protocol-level "monetary policy." Ethereum Improvement Proposals ethereum.org

5) Why cross-border transfers often feel slow/expensive vs. blockchains

- **Many hops & business hours:** messages cross several banks, each with checks and batch windows.
- **Pre-funded jars everywhere:** cash sits idle in multiple countries to make payouts quick at the end-point.
- **On a blockchain:** you swap into a token, **settle on a shared ledger in seconds/minutes**, and swap out at the destination—often with **no pre-funded jars** if the token is liquid in both places (we'll dissect XRP's version of this later). Ripple describes this "on-demand" model explicitly in its payments docs. Ripple documentation

The concept of pegging expanded

1) "Backing" 101 — the treasure-chest picture

Imagine a money issuer with a **treasure chest** in the back room.

- When it **issues** 1 new unit of money, it promises you can **bring that unit back** and get a fixed amount of something in return from the chest (e.g., gold, or U.S. dollars).
- That "something" is the **reserve**. The promise ("you can redeem at a fixed rate") is the **peg**.
- As long as people believe the chest is full enough and redeemable on demand, the money trades at par with the reserve.

This creates an **arbitrage**: if the money slips below par, buy it cheap and redeem for full reserve value; if it trades above par, redeem reserve to mint money and sell it. The peg holds **only if** the issuer truly has adequate reserves and honors redemption. Historically, this was the **gold standard** (redeem notes for gold) and **currency boards** (redeem for a foreign currency like the USD). Encyclopedia Britannica

2) Three real-world "backing" models (how the machine behaves)

A) Commodity standard (classic gold standard)

- **Rule:** each money unit equals a fixed amount of **gold**; you can swap notes for gold at the mint.
- **How market forces behave:** if your country **imports more than it exports**, gold **leaves** to pay foreigners; your **money supply shrinks**, prices/wages fall, and you become more competitive again — the famous **price–specie flow** adjustment. Surplus countries get gold inflows, money expands, prices rise, and competitiveness cools. It's a self-correcting (but often painful) mechanism. Wikipedia

Modern echo: After WWII the **Bretton Woods** system pegged other currencies to the **USD**, and the USD was pegged to **gold at $35/oz**. As global dollars outgrew U.S. gold, the promise buckled; in **1971** the U.S. **ended dollar–gold convertibility**, and the system collapsed. Federal Reserve History +1 Office of the Historian

B) Fixed FX peg / Currency board (backed by another currency)

- **Rule:** your money is fixed to, say, the **USD** (e.g., 7.8 local per $1). The central bank promises to **buy/sell** at that rate **on demand** and holds ample **USD reserves** to prove it.
- **Tightest version (currency board):** you can only issue new local money **when** USD reserves come in; redemption is automatic. **Hong Kong** has run this since 1983, keeping HKD inside a tight 7.75–7.85 band per USD with a rules-based system. Hong Kong Monetary Authority +1

When pressure hits:

- If your currency wants to **weaken**, you **sell reserves** (buy back your money) and/or **raise interest rates** to make holding your money more attractive.
- If it wants to **strengthen**, you **buy reserves** (issue more local money) and/or **cut rates**.
- If reserves or political will run out, the peg can **break** (e.g., **Argentina's** 1:1 USD peg collapsed in 2002). IMF eLibrary IMF

C) Pure fiat (no hard backing)

- **Rule:** the currency is **not convertible** to a reserve asset on demand. Value rests on the state's power (taxes, legal tender laws) and prudent policy (inflation targeting). Most currencies today are fiat. (Background: Bretton Woods' end in 1971 is the watershed.) Office of the Historian

3) Why "backing" dictates market forces (the levers you control)

Think of three dials: **(1) exchange rate**, **(2) interest rate / money supply**, **(3) capital mobility**. You can't freely fix all three at once (the "impossible trinity").

- With a **hard peg/backing**, you're **promising the exchange rate**. So your **interest rates and money supply must dance** to defend it, and you must hold enough **reserves** for redemptions. If belief wobbles, speculators test you; if your chest looks light, they'll run it down. (HK defended successfully by rule + reserves; Argentina couldn't.) Hong Kong Monetary Authority IMF eLibrary
- Under a **gold standard**, trade deficits **drain gold → shrink money → lower prices**; surpluses do the reverse (automatic but often recessionary). Wikipedia
- Under **fiat**, you get **policy freedom** (rates, QE/QT) but must maintain **confidence** to avoid high

- Under **fiat**, you get **policy freedom** (rates, QE/QT) but must **maintain confidence** to avoid high inflation.

4) Where blockchain comes in (today, and what's possible)

A) Fiat-backed stablecoins: "mini currency boards" in the private sector

- **What they are:** digital tokens redeemable 1:1 for dollars/euros held as **reserves** (cash + T-bills). **USDC** publishes regular reserve **attestations**; the peg is defended by **redemption arbitrage** (if USDC < $1, buy and redeem). This mimics a currency board's mechanics on the open internet. Circle +1
- **When pegs wobble:** redemptions surge; good issuers meet them. **Algorithmic** "pegs" without solid backing can **death-spiral** (e.g., TerraUSD in 2022), underscoring why real reserves matter.
  Bank for International Settlements +1

B) Gold-backed tokens: putting the chest on-chain

- **PAX Gold (PAXG)**: each token legally represents **1 troy ounce** of specific LBMA gold in vaults; value tracks physical gold; redeemable. **Tether Gold (XAUt)** similar idea with Swiss-vaulted bars. These are **commodity-backed money-like tokens** with instant, 24/7 transfer. Paxos Tether
- Market is still small versus fiat stablecoins, but PAXG+XAUt dominate tokenized precious metals.
  CoinGecko Mesirow

C) Proof-of-Reserves (PoR): real-time window into the chest

- On-chain **oracles** (e.g., Chainlink PoR) publish automated feeds showing reserves vs tokens outstanding, letting smart contracts **halt minting** or trigger **redemption gates** if collateral falls short. Think of it as a live dashboard that reduces "trust me" to **"verify me"**. Chainlink +1

D) Programmable pegs and future public money

- A central bank or regulated issuer could implement a **programmable currency board**:
  - mint only when reserves increase;
  - auto-pause redemptions if reserves dip;
  - publish reserves/flows on-chain in near-real time.
    This doesn't remove macro constraints (you still need real reserves), but it improves **transparency, speed, and auditability**.

Important reality check: blockchain **can't conjure reserves**. It can *prove* and *move* them faster, make redemption rules automatic, and expose shortfalls early — but the hard economics (export earnings, fiscal policy, credibility) still rule outcomes.

other novel applications

1) Elections

What people hope blockchain can fix

- **Tamper-evident records:** once a result or log is written to a public ledger, changing it later is obvious.
- **Shared timeline:** all parties (parties, media, observers) see the same sequence of events.

Where blockchain really fits (and where it doesn't)

Think of an election as 3 layers:

A. Ballot casting (voters marking choices)

- What technologists dream: "vote on your phone; results go to a blockchain."
- Reality: putting *voting* itself online has unsolved risks (malware on the voter's device, coercion at home, network attacks). Independent experts (e.g., MIT researchers) found serious vulnerabilities in a well-known mobile-voting app and warned that online voting—even with a blockchain back end—can expose elections to undetectable interference. MIT News MIT Internet Policy Research Initiative
- Professional groups caution the same: a blockchain doesn't solve the core dangers of Internet voting (malware/coercion/data-availability). Verified Voting

B. Counting & tabulation

- Good fit: after precincts tally **paper ballots**, publish a **hash** (a digital fingerprint) of each precinct's

results file to a blockchain. Any later change to the file would no longer match the on-chain fingerprint.

- Also good: log **chain-of-custody events** (when boxes sealed/opened, who transported them) as timestamped entries. That creates a public, append-only audit trail *without* moving the secret vote itself onto the internet.

C. Public reporting & audits

- Good fit: publish **final tallies and audit samples** on-chain so anyone can verify what was reported, and that risk-limiting audits were done.

A real attempt (and lessons)

- **West Virginia's 2018 pilot** let certain overseas voters cast ballots via a mobile app using a permissioned blockchain; later, security research highlighted vulnerabilities; critics urged halting online voting. The episode is now a textbook example of *where to draw the line*: keep **ballot casting** on paper + in-person safeguards; use blockchains for **public, tamper-evident posting of results and logs**, not for Internet voting itself. NASS MIT News Oxford Academic

Who pays/earns: election agencies (or donors) pay for the software; blockchains used here are typically public and very low-fee; no "earning" motive.

Bottom line for exams:
Use blockchain for **publishing and auditing** (hashing precinct results, logging custody), *not* for remote Internet voting. Cite the MIT analysis and Verified Voting position to argue this clearly. MIT News Verified Voting

offline Transactions

2) Can blockchains do "offline" transactions?

First, let's be precise about "offline"—it can mean different things:

- **Offline signing (authorization):** you create/sign a transaction while your wallet is offline (air-gapped laptop, hardware wallet).
- **Offline transport:** you carry the signed transaction on a USB/QR and hand it to someone (still not on the network).
- **Offline *final settlement*:** value changes hands **even if nobody is connected to the blockchain** at that moment.

Now, what's real today:

A) Offline signing = YES (common)

You can sign safely on an offline device, then later **broadcast** the transaction when *anyone* gets connectivity. But until a node sees it and miners/validators include it in a block, **nothing settled**. Your course notes show new transactions must be **broadcast**, collected into the **mempool**, and then into a block; if a transaction isn't propagated or included, it can even **fall out** of mempools and must be resent (often with a higher fee). That's why *connectivity at some point* is non-negotiable. CC25-3 (1) CC25-3 (1)

B) "True offline cash-like transfer" on a public chain = Not trustless

Why: the network prevents **double-spends** by maintaining a single, global order of transactions. If Alice and Bob are *both* offline, Alice could copy the same coins and pay Bob *and* Charlie. Only when a node connects and checks the global ledger can the double-spend be caught. Without some **extra trust or secure hardware**, you can't have finality offline.

Designs you'll see (and what they trade):

- **Secure-element cards / hardware wallets that decrement an internal balance** and prevent cloning; devices later sync to the chain. Works only if you trust the hardware/vendor; it's not purely trustless

blockchain security.

- **Chaumian e-cash / custodial tokens:** a mint (issuer) gives you bearer tokens you can pass around offline, then someone redeems them later. Fast and private, but you now trust the **issuer** (and it's no longer decentralized settlement).
- **CBDC offline modes:** similar idea in central-bank pilots—secure chips or "smart cards" swap value locally, then reconcile later. Useful, but the **trust** sits with the central bank/device security, not with a public blockchain.

C) "Almost-offline" delivery networks = YES, with caveats

You can route transactions through **alternative links** (mesh radios/SMS/satellite uplinks) to reach at least *one* online node. That's not *offline settlement*—it's **delay-tolerant connectivity** so the chain still confirms your payment.

short answer: yes—using "lower quality" links (satellite downlink, mesh radios, SMS/USSD, ham radio) can be a **real advantage** over traditional digital payments in places with poor or censored internet, or during outages/disasters. blockchains don't care *how* your signed transaction reaches the network; they only care that it **eventually** reaches one online node, and that you can **receive** fresh blockchain data (at least headers) to know what's final. that makes payment rails you can run over almost any channel.

blockstream.github.io

below are two complete, practical flows (whiteboard-ready), then a crisp "why this can beat traditional rails" and the trade-offs.

Flow 1 — "No internet here, but we still accept Bitcoin" (satellite downlink + SMS/mesh uplink)

Actors: a shop in a connectivity-poor area, a customer with only basic signal (SMS or a short-range mesh), and the wider Bitcoin network.

What the shop sets up (once):

1. Mount a small dish + USB receiver. It listens to **Blockstream Satellite**, which **broadcasts the Bitcoin blockchain worldwide 24/7**. That keeps the shop's node fully in sync **without any internet** (receive-only). If the ISP is down or censored, the node still sees every new block. blockstream.com blockstream.github.io
2. (Optional) A tiny "gateway" box that can forward **outgoing** messages when *some* burst of connectivity exists (even intermittently).

A payment, step by step:

1. **Customer signs** a small (≈200–400 bytes) payment on their phone/wallet. (A typical Bitcoin tx really is just a few hundred bytes.) Learn Me A Bitcoin
2. **Uplink via anything:** the customer's app hands that signed tx to (a) an **SMS** gateway, or (b) a **mesh radio** hop (e.g., goTenna). That message is relayed until it reaches **one** internet-connected node, which gossips it to the network. (This "TxTenna" pattern is already deployed: mesh → an online user → Bitcoin network.) WalletScrutiny GitHub
3. **Merchant verifies:** their satellite-synced node immediately sees the mempool add the tx (via its own peers) and, soon after, sees it **confirmed in a block** beamed down by satellite. Because the satellite is a **one-way downlink**, it's great for **receiving** blocks even if local internet is broken; you still needed that tiny uplink in step 2 to **send**. blockstream.github.io Bitcoin Stack Exchange
4. **When to hand over goods:** for small items the shop may accept on mempool/zero-conf risk; for larger items, wait 1+ confirmations (≈10 minutes per block). If you want instant finality-like UX, use **Lightning** (see Flow 2).

Why this is an advantage over traditional rails:

- If the ISP is out (floods, censorship), the shop's node **still receives the chain** via satellite, and the customer can still **uplink** via SMS/mesh/ham radio—there's no central switch to be up. Traditional systems typically **depend on live, bidirectional internet** to a central operator (bank switch, card

network). blockstream.com blockstream.github.io

Flow 2 — "Feature phone, no data" (Lightning via USSD/SMS)

Actors: a customer with a feature phone, a USSD/SMS service (custodial), the Lightning Network, and the merchant.

How it works in practice:

1. Customer dials a **USSD code** (e.g., `*920*8333#`) and chooses "Pay." Services like **Machankura** map phone numbers to Lightning addresses. 8333.mobi Gate.com
2. Behind the scenes, the provider's node sends a **Lightning payment** to the merchant's Lightning invoice. This routes over internet-connected Lightning nodes; the customer only used **USSD/SMS**. CoinDesk
3. The merchant gets **instant settlement** on their Lightning wallet; no block wait. (The provider later rebalances channels as needed.)

Why it's compelling:

- Works on **basic phones** and **low-signal** environments. India's UPI also has an offline feature-phone mode (123PAY), but that still depends on NPCI/bank connectivity in real time; here the **telco + Lightning node** path is enough, and it can even traverse international routes without correspondent banks. NPCI

Caveat: this is **custodial**—you trust the USSD operator with your funds (like trusting a wallet company). Non-custodial Lightning over radio/mesh is possible but more complex; still, experiments have sent Lightning payments **over ham radio** to prove the transport-agnostic model. CoinDesk Max Fang's Notes

Why this can beat traditional digital payments (when the network is fragile)

1) Works over almost any channel

- Bitcoin/Lightning transactions are tiny, so you can **send** them via SMS, mesh, or ham radio; you can **receive** the chain via **satellite** even if local internet is blocked. That gives **coverage and censorship-resilience** traditional systems lack. Learn Me A Bitcoin blockstream.com

2) Fewer single points of failure

- Card networks and bank switches are centralized services; if the switch or your bank's API is down, you're stuck. A blockchain node plus satellite downlink keeps you updated with no central uplink required; all you need is **some** path to get your signed tx to **one** online node. blockstream.github.io

3) Disaster & remote-area readiness

- In disasters, mesh radios or SMS gateways may come back **before** broadband. You can resume commerce with delay-tolerant links; confirmations arrive by satellite. There are documented community setups using **goTenna meshes** and **Blocksat** for exactly this contingency. WalletScrutiny blockstream.github.io

4) Cross-border without correspondent chains

- A Lightning/Bitcoin payment reaches a foreign merchant **without** the multi-bank message dance. (By contrast, UPI is superb domestically and has some cross-links, but generally rides bank/bilateral infrastructure.) NPCI

...and the trade-offs (be clear-eyed for exams)

- **Latency & UX:** radio/SMS/mesh can be **slow** or intermittent. Great for resilience, not for flashy POS unless you use **Lightning** (instant) or accept small zero-conf risk. Bitcoin Magazine
- **Trust model:** USSD/SMS Lightning wallets are usually **custodial**; you're trusting an operator. Purely self-custodial + offline-ish setups are possible but take more gear/know-how. 8333.mobi
- **Satellite is downlink-only:** it keeps you synced but **can't broadcast** your tx by itself; you still need *some* uplink (SMS/mesh/radio/brief internet). Bitcoin Stack Exchange
- **Compliance/ramps:** you still need local on/off-ramps (exchanges, merchants) and to follow local rules.