Lecture 1

Introduction/Course Logistics/Mental Models

- goals for lecture:
  - mental model for blockchain protocols and web3
    * what does this tech achieve that we didn't already have?
    * cf., mental model for Internet (near-instantaneous global digital communication)
    * general-purpose functionality of a computer, but with the "decentralization/ownerlessness" of the Internet
  - overview of course and its requirements
- mental models for blockchains/web3:
  - general-purpose functionality of a computer (like your laptop)
  - "decentralization" of the Internet (i.e., result of coordination by many diverse parties, no one owner or operator)
  - Internet: shared global infrastructure for *communication* (i.e., data dissemination)
  - blockchain protocol: shared global infrastructure for *computation* (i.e., data processing) [a.k.a. the "computer in the sky"]
  - so can think about a blockchain as either:
    * like your laptop, but everyone else also has an account, and with the hardware swapped out for the Internet
    * the Internet, with a (virtual) shared computer sitting on top
- example application: ownership of digital assets
  - (in a much more general sense than mere cryptocurrencies)

  - in general, the same way a computer networking course doesn't talk about social media platforms, we will mostly won't discuss applications and focus instead on how to build the infrastructure that enables those applications
  - but digital assets is a fundamental example and demonstrates why new technology is needed
  - "ownership" = think in the same sense as your physical possessions, for stuff you bought or created in the digital realm (e.g., a concert ticket, or social media content)
  - viscerally felt gap between strength of owndership for physical and digital possessions; blockchain technology can help narrow this gap

- next: elaborate on analogies between blockchain protocols and two other "intermediate layers," operating systems and the IP protocol

- cartoon of a computer:

  - visualize as a stack of layers, with user-facing layer at top
  - top layer: applications like Photoshop, Word, etc.
  - bottom layer: hardware/physical machine
  - intermediate layer: operating system
  - roles of the OS: (will be strong parallels with blockchain protocols)
    * acts as a "master program" that coodinates all applications (e.g., which program is using the processor at any given time)
    * provides a "virtual machine" abstraction to applications (with an imaginary processor, memory, etc.) — programmers can code *as if* their program will run (by itself) on a physical realization of the virtual machine
  - thus, separation of responsibilities: developers are responsible for translate what a user wants (e.g., "crop an image") into a sequence of VM instructions (typically in two stages, first in a high-level language like Java and outsourcing the rest to a compiler), the OS is responsible for mapping VM instructions to machine instructions (which are realizable by the actual hardware)
  - note: the OS insulates applications from hardware, and vice versa (recurring theme of an "intermediate layer")
    * can innovate on applications without worrying about the underlying hardward, as long as you conform to the OS's virtual machine
    * can innovate on hardware without worrying about applications, as long as you can realize whatever the OS might ask of you
  - recall litmus test: is this tech already good enough to realize the dream of property rights for digital assets?

    * good news: capable of any computation (e.g., certainly capable of maintaining a logical spreadsheet that tracks who own what)
        - aside: T. J. Watson (President of IBM) in 1943: "world market for a total of 5 computers"
    * bad news: neither shared nor decentralized
        - for this reason, not suitable for ownership of digital assets
        - computer could crash, go offline
        - anyone with admin privileges could change content or prevent updates, etc.

- cartoon of the Internet:
  - again, top layer is application layer, user-facing (send an email, load a Web page, etc.)
  - again, bottom layer is hardware/physical devices (which actually physically transport 0s and 1s from one place to another)
  - intermediate layer here is IP (Internet Protocol), whose responsibility is to provide point-to-point communication to layers above
    * like a digital version of the postal service—try to transport a data packet from a machine with a given IP address to the machine with the packet's destination IP address
  - local networks are then responsible for getting (by whatever means) a packet to traverse one hop
  - applications can use IP functionality without worrying about how it's actually realized (cellular, WiFi, Ethernet, etc.)
  - lower-level networks can innovate on how they get 0s and 1s from one place to another, as long as they can realized the one-hop routing functionality required of them by IP
  - decoupling of the application and hardware layers particularly striking in the Internet example: IPv4 has been around since the 1980s, unchanged, even as massive technological leaps have happened at the application and hardware layers (Web, cloud, mobile, social, etc.)
  - recall litmus test: is this tech already good enough to realize the dream of property rights for digital assets?
    * good news: shared and decentralized (everyone uses same network, no one owner or operator, very difficult to shut down or obstruct use due to the large number of independent parties involved)
    * bad news: only passes bits around, "stateless" by design (e.g., not designed to track anything, digital assets or otherwise)

- cartoon of web3:
  - one way to think about it: a shared computer but with its hardware swapped out for the Internet
  - another way: put a virtual machine on top of the Internet
  - can again visualize as a three-layer stack:
    * top layer: user-facing applications (in this context, sometimes called "smart contracts"), e.g. Uniswap, OpenSea, Farcaster, etc.
    * bottom layer: the Internet (in effect, Web3 infrastructure piggybacks on existing Internet infrastructure)
    * intermediate layer: blockchain protocol (e.g., Ethereum, Solana, etc.)
  - strong parallels between the role of a blockchain protocol and that of the operating system of a computer:
    * acts as a "master program" that coordinates all applications/smart contracts (e.g., determines which one is currently be executed by the virtual machine)
    * exports a VM to application/smart contract developers (though in this case, the VM is not meant to be an abstraction of any particular physical machine)
    * programmers (e.g., in Solidity or Rust) can code applications as if their program would be run a physical machine corresponding to the exported VM
  - like the Internet, the product of coordination between many physical machines (ideally with different owners/operators)—in this sense, "decentralized"
  - analogy: Web servers are physical machine all over the planet that coordinate by running a common protocol (http) to enable end users to experience the Web; physical machines running a common blockchain protocol (e.g., Ethereum) coordinate to enable end users to experience what is logically a shared general-purpose computer
  - note, fairly meta (confusing to non-CS folks): a collection of physical machines coordinate (using a blockchain protocol) to simulate the results of a computation if it were, hypothetically, to be run a physical realization of the virtual machine
  - why take many physical computers and use them to simulate only one? only the latter is decentralized and, e.g., suitable for attesting to ownership of digital assets

- high-level syllabus:
  - goal of the course is to learn how to build the "computer in the sky" (in the same sense that you learn what it takes to build the Internet in a computer networking course, though in our case the technology is still very much in flux)
  - will have three parts (each roughly 9 lectures):

- Part I: how to build a shared global virtual computer without regard to performance (e.g., power of a 1950s computer) and with "permissioned" infrastructure (e.g., a fixed and known set of 22 or 100 physical machines, all over the world)
  * from day one, developing and using applications will be permissionless (i.e., open to anyone) — absolutely central to the blockchain/Web3 vision
  * this part is largely a selection of the most relevant material from classic computer science courses (which not everyone takes, alas), like distributed systems/computing and operating systems
  * topics include: fault-tolerant consensus, virtual machine execution
- Part II: focus on performance/scaling (while retaining the "permissioned infrastructure" assumption); e.g. would like our virtual machine to at least have the processing power of a 1990s or 2000s-era computer
  * for much of the material, one could imagine a world in which it would already be taught in existing computer science courses (even in the pre-blockchain era), but for the most part it's not—the design of high-performance blockchain protocols seems to be forcing a novel synthesis of a number of known concepts
  * topics include: rollups (optimistic and "zk") and sequencers, SNARKs, light clients, bridges, data availability, transaction fee mechanisms, etc.
- Part III: permissionless protocols: an even harder version of the problem in the physical machines running the protocol can join and leave as they wish (as opposed to being fixed and known)
  * would've seemed crazy in the pre-Bitcoin era (still seems a bit crazy/impossible, tbh), yes most of today's major blockchain protocols (Bitcoin, Ethereum, Solana, etc.) are to some degree permissionless
  * topics include: proof-of-work vs. proof-of-stake (approach to sybil-resistance), incentives, public mempools, MEV, etc.

- comments on the course:
  - course is about a new computing paradigm, not digital money
    * despite what you may have heard, blockchains $\neq$ cryptocurrencies
    * equating the two is like equating the Internet with email (the former is a general-purpose technology, the latter is one very specific thing you can do with the technology)
    * cryptocurrencies not really relevant for us until Part III of the course, but even there, they are a means rather than and ends (useful to provide incentives, pay transaction fees, use as collateral, etc.)
  - principles over protocols

    * e.g., when you take an operating systems course, you focus on principles and design trade-offs that relevant for most or all OSes, and use specific OSes (Android, etc.) as case studies or to illustrate the more general concepts
    * same thing here: you'll learn plenty about Bitcoin and Ethereum (and several other "layer-1s" and "layer-2s") as we go along, the emphasis is on fundamental design choices that any blockchain protocol designer must grapple with, and will view specific projects through the lens of these general principles
  - a new area of computer science
    * forming literally in real time, before our eyes
    * synthesizes existing parts of computer science (distributed systems, cryptography, etc.) and other fields (e.g., economics and game theory) but now clearly an intellectually deep area in its own right
    * this course is an inadvertent capstone course—you will see many threads of your previous CS education come together in surprising and satisfying ways
    * your opportunity to get in on the ground floor of this area, like getting into the Internet/Web in the early 1990s
    * this course is your one-stop shop for jumping into industry or research; master this material and you will have a tremendous competitive advantage (currently, demand for this skill set is much bigger than supply, and outside of a course like the skill set is very difficult to acquire)

- deliverables
  - (50%) open-ended team project, teams of 3 or 4
  - (40%) homeworks, maybe 8 or 9 total
  - (10%) participation
  - no exams