# Ethereum

Ethereum là một mạng máy tính phân tán, mỗi nút trong mạng thực thi một số mã lệnh (được gọi là: Hợp đồng thông minh – Smart Contract ), sau đó lưu trữ trạng thái kết quả trong một chuỗi khối (blockchain). Nhờ vào các đặc tính của blockchain đại diện cho trạng thái ứng dụng dẫn đến các ứng dụng có thể chạy chính xác mà không thể bị ngừng hoạt động, kiểm duyệt, lừa đảo hoặc can thiệp của bên thứ ba.

Giống như các mạng blockchain khác, Ethereum có một loại tiền điện tử gọi là Ether (ETH). ETH là tiền kỹ thuật số. Nó hoàn toàn là kỹ thuật số và có thể được gửi đến bất cứ ai trên thế giới ngay lập tức. Việc cung cấp ETH không được kiểm soát bởi bất kỳ chính phủ hoặc công ty nào - nó được phân cấp, và hữu hạn Mọi người trên khắp thế giới sử dụng ETH để thực hiện thanh toán, như một kho lưu trữ giá trị hoặc làm tài sản thế chấp.

Nhưng không giống như các mạng blockchain khác, Ethereum có thể làm được nhiều hơn thế. Ethereum có thể lập trình được, có nghĩa là các nhà phát triển có thể sử dụng nó để xây dựng các loại ứng dụng mới trên nền tảng công nghệ blockchain.

Các ứng dụng phi tập trung này (còn gọi là Dapps) sử dụng hiệu quả những lợi ích của tiền điện tử và công nghệ blockchain. Chúng rất đáng tin cậy, có nghĩa là một khi ai đó tải bất cứ thứ gì của họ lên Ethereum, chúng sẽ luôn chạy như được lập trình. Chúng ta có thể kiểm soát tài sản kỹ thuật số để tạo ra các loại ứng dụng tài chính mới. Chúng có thể được hoạt động phân tán phân tán mà không chịu điều khiển từ bất cứ người hay tổ chức nào .

Ngày nay, hàng ngàn nhà phát triển trên toàn thế giới đang xây dựng các ứng dụng trên Ethereum và phát minh ra các loại ứng dụng mới :

* Ví tiền điện tử cho phép bạn thực hiện thanh toán giá rẻ, tức thì bằng ETH hoặc các tài sản khác.
* Các ứng dụng tài chính cho phép bạn vay, cho vay hoặc đầu tư tài sản kỹ thuật số của bạn.
* Các thị trường phi tập trung (Decentralized markets), cho phép bạn giao dịch các tài sản kỹ thuật số.
* Trò chơi mà bạn sở hữu tài sản trong trò chơi có thể kiếm được tiền thật
* Và nhiều ứng dụng khác nữa.

Không có công ty hoặc tổ chức tập trung nào kiểm soát Ethereum. Ethereum được duy trì và cải thiện theo thời gian bởi một cộng đồng trên  toàn cầu, những người làm việc trên mọi thứ từ giao thức cốt lõi đến các ứng dụng tiêu dùng.

# Smart Contract

Một “Smart Contract” là thực chất là những đoạn mã được thực thi trên môi trường phân tán chẳng hạn nền tảng Ethereum. Nền tảng mà nó được thực thi cung cấp cho đoạn mã này một số thuộc tính như: tính bất biến, hoạt động xác định, trạng thái phân tán và có thể kiểm chứng, v.v. Trạng thái được quản lý bởi mã này  lưu trữ bất biến trên blockchain và bất kỳ ai cũng có thể kiểm tra xem hiện tại trạng thái là chính xác bằng cách phát lại tất cả các giao dịch từ đầu chuỗi.

Trong Ethereum, các hợp đồng được cung cấp một địa chỉ để xác định duy nhất chúng (đó là một hàm băm của địa chỉ người tạo và các giao dịch mà họ đã gửi trước đó). Sau đó, người dùng có thể giao tiếp với địa chỉ này bằng cách gửi ether cho nó, gọi những hàm đã được định nghĩa, truy vấn trạng thái phân tán mà nó quản lý, v.v.

## Gas

Hợp đồng thông minh được thực thi bởi mỗi nút đầy đủ đơn nhất trong mạng, Việc này lặp lại các thông điệp để giảm xác suất xảy ra lỗi trong quá trình truyền nhưng mặt khác lại tốn rất nhiều năng lượng và thời gian.

Từ gas là một nhãn trừu tượng được đưa ra đại diện cho chi phí thực hiện tính toán. Ngoài ra, còn có một giới hạn gas do mạng thực thi, để giải quyết vấn đề tạm dừng; tức là, bạn không thể viết một chương trình không bao giờ kết thúc, bởi vì khi hết gas, tính toán sẽ bị mạng từ chối.

Gas là chi phí được trả cho các miner nhằm khuyến khích các họ đưa các giao dịch vào trong block và lưu trữ trên mạng blockchain.

Hai  thành phần của gas là:

* Gas limit( giới hạn gas) : là lượng gas tối đa được dành cho việc tính toán.
* Gas price (giá gas) : đơn vị giá của gas ( GWEI) . 1GWEI = 109WEI ( tương tự như ETH). Mặc định Gas price = 20GWEI.

## Các ứng dụng phân tán -Distributed App (dApp)

Về phía server các dApp là đơn thuần là những smart contract được lưu trữ trên mạng Ethereum.

Ứng dụng phân tán không cần lưu trữ tất cả trạng thái của nó và thực hiện tất cả các tính toán của nó trên blockchain, điều này sẽ khá tốn kém. Nhưng một ứng dụng phân tán cuối cùng sẽ lưu trữ trạng thái đáng tin cậy trên blockchain Ethereum, mà bất kỳ ai cũng có thể sau đó đọc. Nhiều ứng dụng phân tán như IPFS và Golem cũng tính toán và lưu trữ  phi tập trung.

### a, Ứng dụng phân tán phía người dùng-dApp Client

Các ứng dụng phân tán này thường đi kèm với một số loại giao diện thân thiện với người dùng (frontend) vì không ai muốn gửi giao dịch đến (/ từ) các hợp đồng bằng cách sử dụng một yêu cầu thủ công với băm và các dòng lệnh phức tạp.

Những ứng dụng khách này thường được viết bằng JavaScript , Go, Nodejs,…

### b, Trình duyệt ứng dụng phi tập trung-dApp Browsers

dApp Browsers là một ứng dụng giúp việc sử dụng các ứng dụng dApp (thường là javascript có giao diện với nút Ethereum để giao tiếp với các hợp đồng thông minh) dễ dàng hơn.

Hai mục đích chính của dApp Browsers là :

* Cung cấp kết nối tới nút Ethereum (được lưu trữ cục bộ hoặc từ xa) một cách dễ dàng.
* Và cung cấp giao diện tài khoản (ví điện tử- wallet) cho người dùng để họ có thể dễ dàng giao tiếp với các ứng dụng này.

## Ethereum Nodes

Nút lưu trữ một bản sao của blockchain và tùy chọn thực hiện tất cả các giao dịch để xác nhận tình trạng kết quả. Một nút Ethereum có thể là full node hoặc một nút client nhẹ (như Geth hoặc Party) hoặc sử dụng bên thứ ba như Infura hay cũng có thể là một nút local để test và dev.

## Tương tác với smart contract

Cách thông thường để tương tác với smart contract là kết nối với một nút Ethereum và gọi đến hàm của smart contract thông qua JSON RPC API.

Đối với các lập trình viên, có nhiều nền tảng hỗ trợ cách thức giao tiếp với smart contract trên mạng Ethereum khác nhau. Với Javascript có ba thư viện phổ biến là: web3.js, ethjs và ethers.js . Với GO, có go-ethereum.

Khi “deploy” một smart contract, thực chất là việc gửi một giao dịch tới một địa chỉ nào đó với tham số là bytecode của contract.

Có một số ngôn ngữ được xây dựng để viết ra các smart contract ( vd Solidity) sau đó được biên dịch sang bytecode. Có nhiều Framework hỗ trợ viết , triển khai và test contract như :Truffle (javascript), Embark(javascript), Perigord (Go), Populous (Python).

## “Tài khoản” và “ví”

Tài khoản ( account) về cơ bản là một cặp private key và address dùng để lưu trữ một lượng ether mà một người sở hữu. Tài khoản có thể được tạo hoàn toàn miễn phí. Mọi giao dịch trên mạng ethereum đều bắt nguồn từ tài khoản, hợp đồng không có khả năng khởi động các giao dịch.

Ví  Ethereum là một trong hai thứ: hoặc là một giao diện giúp người dùng tạo và gửi giao dịch sử dụng tài khoản của mình, hoặc là một hợp đồng thông minh để gửi và nhận ether.

## EVM và cách tạo ra hợp đồng thông minh

Code của hợp đồng thông minh chạy trên mỗi nút sẽ được thực thi trong môt máy ảo Ethereum (EVM- Ethereum Virtual Machine). Máy ảo này sẽ chạy một số mã bytecode để thực thi hợp đồng thông minh. Tuy nhiên việc viết hợp đồng thông minh trên những mã bytecode như vậy rất khó khăn và tốn thời gian; vì thế nhiều ngôn ngữ bậc cao hơn đã xuất hiện để hỗ trợ việc viết một hợp đồng thông minh trên Ethereum trở nên đơn giản hơn.

Ngôn ngữ phổ biến nhất phải kể đến chính là solidity kế đến là LLL, Serpent, Vyper, Bamboo,…

## Bảo mật hợp đồng thông minh

Khi một hợp đồng thông minh được triển khai cho Ethereum, nó là bất biến và tồn tại mãi mãi. Nếu hợp đồng có dù chỉ một lỗi, bạn không thể gỡ bỏ được hợp đồng đã triển khai mà chỉ có thể triển khai một phiên bản khác của bản hợp đồng đó khi đã sửa lỗi.

Có nhiều loại bẫy liên quan đến bảo mật khác nhau mà code hợp đồng thông minh có thể rơi vào, cả ở cấp độ ngôn ngữ và từ góc độ logic cấp cao. Khi triển khai một hợp đồng thông minh xử lý với tiền thật, bạn cần phải chắc chắn 100% hoạt động của hợp đồng.

# IPFS

## Cơ bản về IPFS

### IPFS là gì?

IPFS là một hệ thống phân tán để lưu trữ và truy cập file, website, application, data. Khi truy cập một website hay tải một file thay vì phải truy cập lên máy chủ của site đó để yêu cầu truy cập, nếu dùng IPFS có thể yêu cầu hỏi tất cả các máy trong IPFS xem đã có thông tin ấy chưa để chia sẻ thông tin đó cho mình. Máy tính của mình cũng có thể đóng vai trò là máy phân tán nội dung cho các máy khác.

### Ý tưởng của IPFS

  IPFS dựa trên ý tưởng sở hữu và tham gia, nơi nhiều người sở hữu các tệp tin khác của nhau và tham gia vào mạng để làm chúng có sẵn. Điều này có nghĩa là IPFS chỉ hoạt động tốt khi nhiều người tham gia và hoạt động trong mạng.

### Lý do sử dụng IPFS

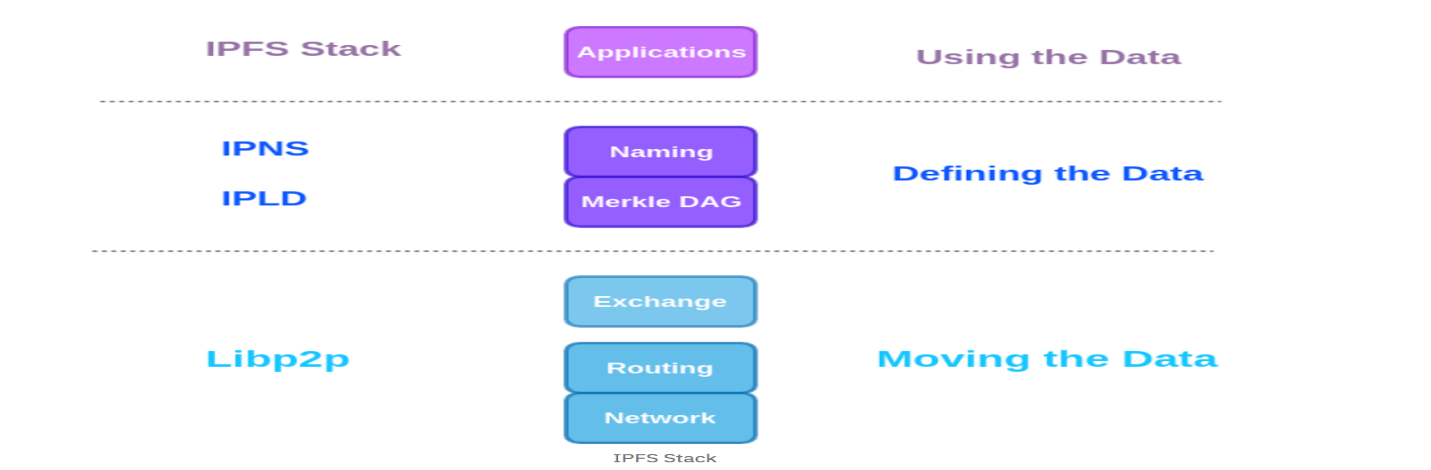
* Supports a resilient internet: có khả năng chịu lỗi, ví dụ như kẻ tấn công vào webserver của mình hay là quản trị viên làm mất data thì chúng ta vẫn có thể truy cập vào máy khác để lấy nội dung.
* Khó khăn trong việc kiểm duyệt nội dung: bởi vì file được lưu trữ tại rất nhiều máy trên thế giới vì vậy rất khó để bất kỳ cá nhân, tổ chức có thể cấm và quản lý nó.
* Can speed up the web when you’re far away or disconnected: Có thể lấy 1 file gần bạn thay vì phải phải truy cập cách xa hàng trăm hay hàng nghìn km, giúp lấy dữ liệu nhanh hơn.

### Content identifiers (CID)

Không chỉ ra vị trí lưu trữ file nhưng nó được tạo ra từ nội dung của file đó. CID không phụ thuộc vào kích thước của nội dung cần lưu trữ. Nó được tạo ra bằng cách băm nội dung.

Bất kỳ sự khác nhau trong nội dung sẽ tạo ra một CID khác nhau và cùng 1 nội dung được thêm vào 2 node IPFS khác nhau, sử dụng cùng cấu hình sẽ sinh ra chính xác một CID

## Cấu trúc IPFS

****

Tầng Application: Nhận và hiển thị dữ liệu

IPNS (inter-planetaty name system) là hệ thống tạo và cập nhật liên kết  nội dung content IPFS. Khi thêm một nội dung lên trên mạng IPFS, dựa vào nội dung của nó, hệ thống sẽ định danh nội dung bằng một giá trị CID.  CID không phụ thuộc vào kích thước của nội dung cần lưu trữ. Nó được tạo ra bằng cách băm nội dung. Bất kỳ sự khác nhau trong nội dung sẽ tạo ra một CID khác nhau và cùng 1 nội dung được thêm vào 2 node IPFS khác nhau, sử dụng cùng cấu hình sẽ sinh ra chính xác một CID. Người dùng sẽ phải sử dụng giá trị CID này để truy cập tới nội dung trên. Tuy nhiên, giá trị này có một vài nhược điểm như sau:

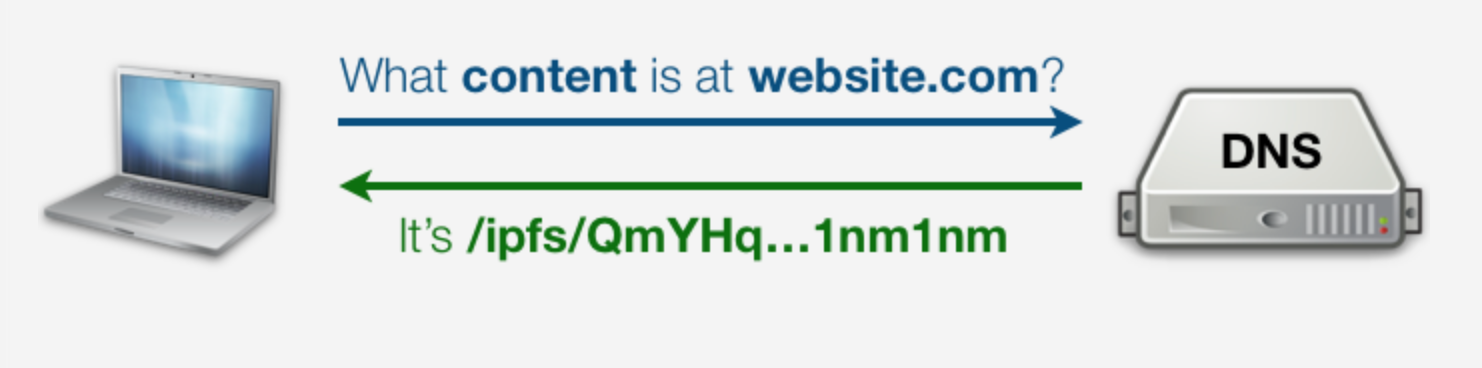
* Khó khăn để có thể đọc và ghi nhớ
* CID có thể thay đổi bất kỳ lúc nào khi content thay đổi. Nếu như người dùng không biết CID mới này thì họ vẫn truy cập vào nội dung cũ.

Để giải quyết vấn đề này IPFS sử dụng IPNS. Thông qua IPNS, người dùng có thể tạo ra một link mới mà được liên kết tới phiên bản mới nhất của nội dung

Tên trong IPNS là giá trị băm của public key, nó được liên kết tới một bản ghi chứa thông tin về giá trị băm hash mà nó liên kết đến và được ký bởi người có private key tương ứng. Một bản ghi mới được ký và xuất bản bất cứ lúc nào. Address của IPNS trông như sau:

/ipns/QmSrPmbaUKA3ZodhzPWZnpFgcPMFWF4QsxXbkWfEptTBJd

IPNS không phải là cách duy nhất để tạo địa chỉ thay đổi trên IPFS. Chúng ta có thể sử dụng DNSLink (hiện nay nó nhanh hơn so với IPNS và tên dễ đọc và dễ nhớ hơn). DNSLink là protocol  liên kết content với dịch vụ DNS.

****

Với DNSLink, có thể lưu trữ link sử dụng bất kỳ tên miền DNS nào. Đầu tiên, đặt link trong bản ghi TXT tại một tên miền phụ xác định. Sau đó có thể phân giải tên miền này tại bất kỳ chương trình nào bằng cách tìm kiếm giá trị trong bản ghi TXT.

**Merkle DAG**: Directed Acyclic Graph (DAG) là một loại biểu đồ trong đó các cạnh có hướng và chu kỳ không được phép. Ví dụ về một DAG, A->B->C ở đó B là con hoặc hậu duệ của A, A có một liên kết với B. Ngược lại A là con của  cha. Một node không phải là con của bất kỳ nút nào gọi là node root. Merkle-DAG là một DAG trong đó mỗi nút có một mã định danh và đây là kết quả của việc băm các nội dung của nút. Một số điểm cần chú ý:

* Merkle-DAG chỉ có thể được xây dựng từ lá, nghĩa là từ các nút không có con. Node cha  được thêm sau khi node con vì định danh node con phải được tính toán trước để có thể liên kết chúng.
* Merkle-DAG nodes là bất biến.  Bất kỳ thay đổi nào trong một node dẫn đến định  danh nó cũng thay đổi và ảnh hưởng toàn bộ đến DAG, khiển root của DAG cũng thay đổi, về cơ bản sẽ tạo ra một DAG khác.
* Mỗi nút trong Merkle-DAG là gốc của một sub Merkle-DAG và sơ đồ con này được chứa trong DAG gốc.

Định danh một đối tượng dữ liệu (như một nút Merkle-DAG) bằng giá trị băm của nó được gọi là content-address. Do đó,  đặt tên định danh nút là Content Identifier hoặc CID. Dữ liệu của mỗi node là CID của hậu duệ của nó A=Hash(B)→B=Hash©→C=Hash(∅)

Merkle-DAGs là cấu trúc tự xác minh. CID của một nút được liên kết đơn phương với nội dung của payload của nó và của tất cả các hậu duệ của nó. Do đó, hai nút có cùng CID đại diện chính xác cho cùng một DAG. Đây sẽ là một thuộc tính quan trọng để đồng bộ hóa hiệu quả các Merkle-CRDT mà không phải sao chép toàn bộ DAG, như được khai thác bởi các hệ thống như IPFS. Merkle-DAGs được sử dụng rất rộng rãi. Tóm lại  Merkle-DAGs giống với hash tree (Merkle hash).

IPLD: IPLD là một bộ tiêu chuẩn và thực hiện tạo cấu trúc dữ liệu phi tâp trung có thể truy cập và liên kết (IPLD sử dụng Merkle DAG). IPLD gồm 2 thành phần:

* Data — blob of unstructured binary data of size < 256 kB.
* Links — array of Link structures.  Đó là các links to các đối tượng IPFS khác. Mỗi IPLD link lại có 3 phần:
  + Name : tên của Link
  + Hash: hash của đối tượng IPFS được liên kết
  + Size: kích thước tích luỹ của đối tượng IPFS được liên kết, bao gồm tất cả các link theo sau nó.

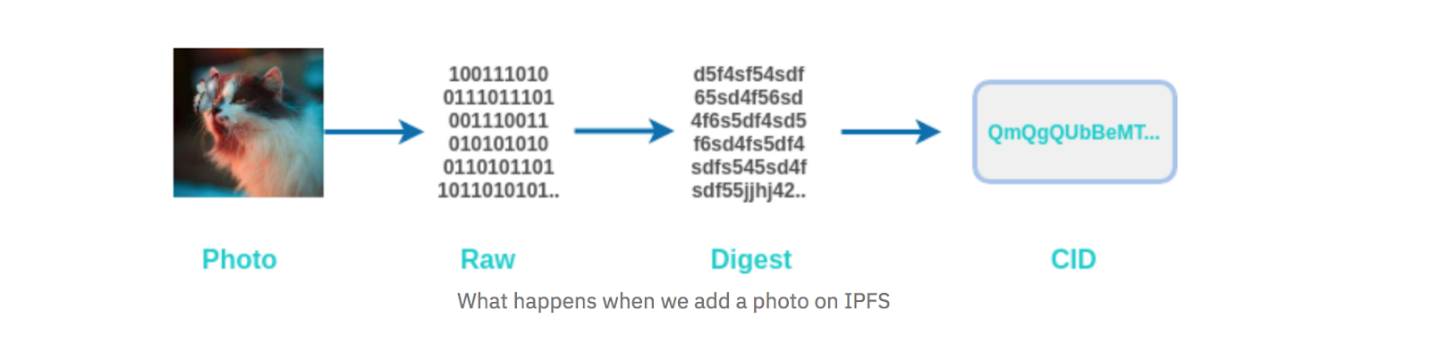
IPFS sử dụng IPLD để quản lý tất cả các block và liên kết chúng lại với nhau.

Libp2p: hiểu đơn giản là tập hợp các protocol peer to peer để tìm các peer, kết nối chúng để tìm content và truyền data. IPFS cho phép bạn cung cấp CID cho nội dung và liên kết nội dung đó với nhau trong Merkle-DAG bằng IPLD. Bây giờ, hãy để di chuyển đến phần cuối cùng: cách bạn tìm và di chuyển nội dung. Để tìm các peer đang lưu trữ nội dung mà bạn muốn (khám phá), IPFS sử dụng  distributed hash table hoặc DHT. Bảng băm là một cơ sở dữ liệu theo dang (key, value). Bảng băm phân tán là một bảng trong đó bảng được phân chia trên tất cả các peer trong mạng phân tán. Để tìm nội dung, bạn hỏi những peers. Libp2p là 1 phần của IPFS, cung cấp DHT và xử lý kết nối, giao tiếp giữa các peer. Khi bạn biết nội dung của mình ở đâu (nghĩa là, peer đang lưu trữ từng block tạo nên nội dung mà bạn quan tâm), bạn sử dụng lại DHT để tìm vị trí hiện tại của các peer (routing). Vì vậy, để có được nội dung, bạn sử dụng libp2p để truy vấn DHT hai lần. Bạn đã phát hiện ra nội dung của mình và bạn đã tìm thấy vị trí hiện tại của nội dung đó - bây giờ bạn cần kết nối với nội dung đó và lấy nó (trao đổi). Để yêu cầu các block  và gửi các block đến các peer khác, IPFS hiện sử dụng một mô-đun có tên Bitswap. Bitswap cho phép bạn kết nối với các peer hoặc peer có nội dung bạn muốn, gửi cho họ danh sách mong muốn của bạn (danh sách tất cả các block mà bạn quan tâm) và yêu cầu họ gửi cho bạn các block bạn yêu cầu. Khi các block đó đến, bạn có thể xác minh chúng bằng cách băm nội dung của chúng để lấy CID.

Tóm lại: Hệ sinh thái IPFS cung cấp CID cho nội dung và liên kết nội dung đó với nhau bằng cách tạo IPLD-Merkle-DAG. Bạn có thể khám phá nội dung bằng cách sử dụng DHT mà libp2p cung cấp và mở kết nối tới bất kỳ nhà cung cấp nội dung nào và tải xuống bằng kết nối đa kênh. Tất cả những thứ này được tổ chức cùng nhau bởi các trung gian của các ngăn xếp, được liên kết, các định danh duy nhất; đó là phần thiết yếu mà IPFS được xây dựng.

## Quy trình hoạt động của IPFS

Khi thêm một dữ liệu dạng ảnh vào IPFS, quy trình diễn ra như sau:



   Ảnh được chuyển thành dữ liệu raw, sau đó hash dữ liệu này được Digest duy nhất. Nếu thay đổi bất kì một bit nào trong photo thì cũng sẽ thu được một Digest hoàn toàn khác. Do vậy có thể chống giả mạo dữ liệu và có thể đảm bảo tính toàn vẹn của dữ liệu, khi có thể kiểm tra xem dữ liệu sau khi tải về có đúng là dữ liệu gốc ban đầu hay không. Sau khi truyền dữ liệu thô vào SHA256 thu được Digest duy nhất, sau đó convert Digest này thành CID (Content Identifier). CID chính là thứ mà IPFS tìm kiếm để nhận lại hình ảnh. Như vậy photo đã được thêm vào IPFS.

Tuy nhiên trong thực tế, khi chúng ta muốn thêm một file có dung lượng lớn vào IPFS, mô hình có thể như sau:

Nếu như file có kích thước lớn hơn 256kb, nó sẽ được chia nhỏ thành các block có kích thước nhỏ hơn hoặc bằng 256kb. Mỗi block sau đó được convert thành CIDs. Để quản lý các block này và liên kết nó với CID, IPFS sử dụng IPLD. Cuối cùng để tìm và lấy nội dung IPFS sử dụng Libp2p.

## Kết nối và lấy data giữa các node

### Cách các node connect với nhau

Khi có một node tham gia vào hệ thống, IPFS sẽ sử dụng hệ thống mật mã khóa công khai để tạo ra 1 cặp khóa public key và private key. Với cặp khóa này các node có thể xác thực lẫn nhau. Trong thực tế IPFS sử dụng hệ thống mã RSA. Để tạo cặp khóa ta chạy lệnh:

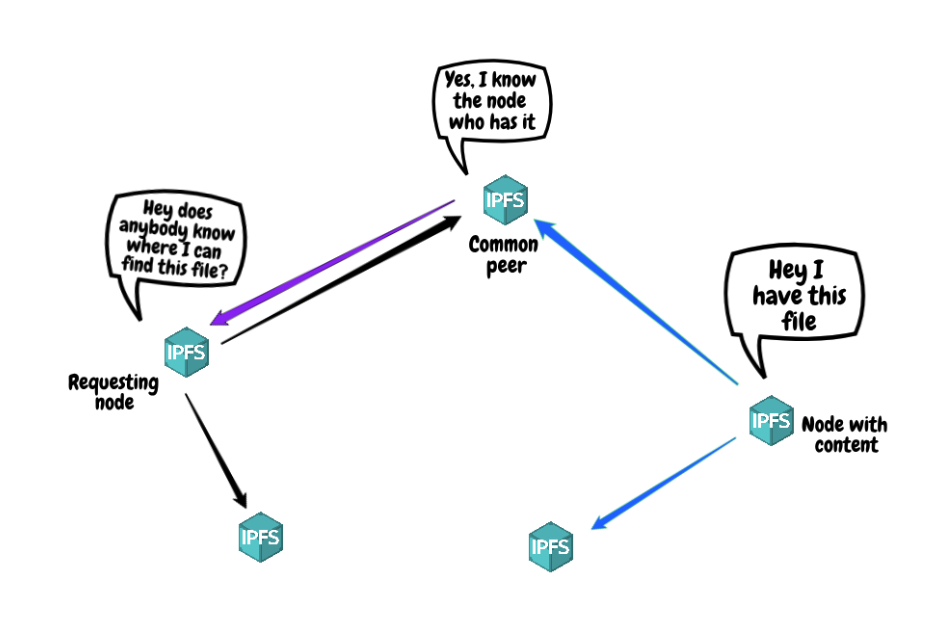
$ ipfs init

Với public key, ta có thể tạo định danh peerId. PeerId được coi như là mã định danh của mỗi một node, từ đó cho phép các node trong mạng có thể xác định danh tính của nhau và tự xác thực khi khi được kết nối.

Khi 2 node kết nối với nhau, chúng sẽ trao đổi public key. Ở đây chúng ta sẽ có sự kiểm tra giữa các node để đảm bảo rằng các peerId khớp với các public key mà 2 node đã trao đổi.

### Cách các node trao đổi nội dung với nhau

Trong IPFS sử dụng DHT(bảng băm phân tán) để hỗ trợ trong định tuyến và truy xuất nội dung của các node trong mạng. Giả sử khi một node A  cần tìm kiếm nội dung với mã hash nội dung đó là CID, node đó sẽ hỏi các node xung quanh. Mỗi node sẽ tra trong bảng DHT của mình và định tuyến dữ liệu của họ cho node A.



## Multiformat

Multiformat project giới thiệu một bộ standards/protocol bao trùm khía cạnh này và cho phép nhiều protocol cùng tồn tại ngay cả khi có một sự thay đổi, hệ sinh thái vẫn hỗ trợ tất cả các version của protocol.

Các multiformat protocol hiện nay:

* Multihash
* Multiaddr
* Multibase
* Multicode
* Multistream
* Multistream-selec
* Multigram
* Multikey

### Multihash

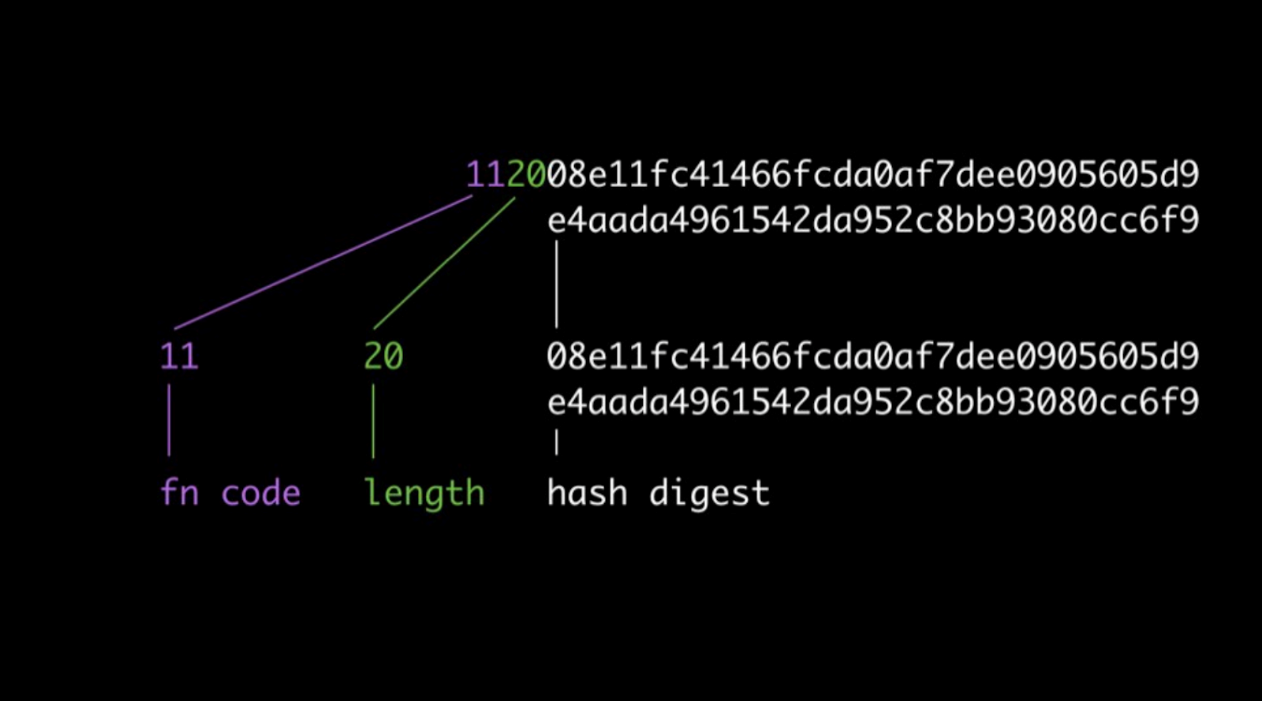
Multihash là một giao thức để phân biệt các đầu ra khác nhau của các function hash. address size và encoding. Điều này cho phép nhiều function hash cùng tồn tại.

Sử dụng multihash đem lại những lợi ích sau:

* Upgrades các function hash một cách dễ dàng và an toàn hơn: ví như một hàm băm không còn an toàn nữa, cần phải thay thế một hàm băm an toàn hơn cho toàn bộ hệ thống, việc sử dụng multihash sẽ giúp upgrades hash function của toàn bộ hệ thống một cách dễ dàng hơn.
* Multihash được thiết kế để upgrades.

Format multihash: theo chuẩn LTV(type-length-value)

* Type <hash\_func\_type> là 1 số nguyên không dấu xác đinh hash function, có một table default quy định type https://github.com/multiformats/multicodec/blob/master/table.csv
* Length <digest-length> chiều dài của digest tính theo bytes
* Value <digest-value> là digest hash function



### Multiaddr

      Multiaddr là một format để encoding address của các network protocol. Nó rất hữu dụng trong các ứng dụng cho phép triển khai nhiều protocol transport và address cùng tồn tại.

# Ứng dụng Smart Contract Ethereum và IPFS vào hệ thống xác thực bằng cấp.

Như đã biết thì blockchain là một công nghệ cốt lõi để xây dựng những ứng dụng phi tập trung đảm bảo tính an toàn và minh bạch của dữ liệu, những dữ liệu được đưa lên blockchain thì không thể bị xoá bỏ hoặc chỉnh sửa. Lợi dụng đặc tính đó, các hệ thống xác thực và truy xuất nguồn gốc ra đời đem lại sự tin  cậy, minh bạch cho dữ liệu đồng thời cũng giúp tiết kiệm nhiều chi phí cho các công việc liên quan đến xác thực.

Chúng em đề xuất ứng dụng công nghệ blockchain cho quy trình xác thực và truy xuất nguồn gốc bằng cấp thông qua hai bước xác thực sau:

* Xác minh bằng cấp thông qua việc truy xuất thông tin bằng cấp được lưu trữ trên blockchain.
* Truy xuất nguồn gốc bằng cấp thông qua việc truy xuất dữ liệu học tập của sinh viên (được lưu trữ trên blockchain) để đạt được bằng cấp và kiểm tra điều kiện hợp lệ thông qua dữ liệu chương trình đào tạo (được lưu trữ trên blockchain)

## Quy trình lưu trữ và xác thực dữ liệu được diễn ra như sau:

Mỗi trường đại học sau khi tham gia hệ thống sẽ được cấp cho một tài khoản Ethereum và một hợp đồng thông minh tương ứng để đưa dữ liệu lên blockchain.

Đối với dữ liệu chương trình đào tạo:

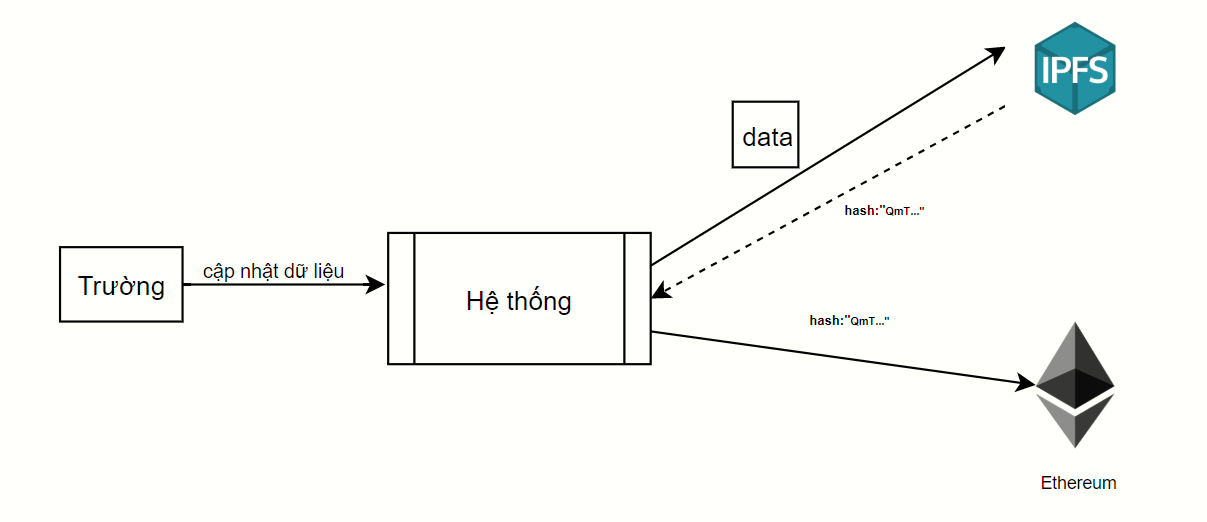
* Các trường đại học cập nhật dữ liệu chương trình đào tạo lên hệ thống.
* Mỗi chương trình đào tạo sẽ được đẩy lên hệ thống IPFS dưới dạng file JSON. Hệ thống IPFS sẽ trả lại 1 mã hash tương ứng với dữ liệu chương trình đào tạo vừa đẩy lên. Mã hash này được ghi lên blockchain thông qua hợp đồng thông minh và tài khoản của trường học.

Đối với dữ liệu học tập của sinh viên

* Sau mỗi kỳ học các trường đại học cập nhật dữ liệu các môn học và điểm của sinh viên lên hệ thống,
* Dữ liệu mỗi môn học của sinh viên được đưa vào 1 file JSON và đưa lên IPFS. Mã hash trả về được lưu lại trên blockchain thông qua smart contract và tài khoản của trường. Các mã hash của cùng 1 sinh viên được lưu vào một danh sách được ánh xạ bởi id của sinh viên tương ứng.

Đối với dữ liệu bằng cấp của sinh viên

* Sau mỗi kỳ trường cấp bằng cho sinh viên, dữ liệu bằng cấp của sinh viên sẽ được đưa lên hệ thống.
* Dữ liệu bằng cấp sẽ được đưa vào một file JSON và đẩy lên IPFS. Mã hash trả về được lưu trữ lên blockchain thông qua smart contract của trường học. Mỗi mã hash tương ứng sẽ đính kèm với một id của sinh viên sở hữu dữ liệu bằng cấp đó.



## Quy trình xác thực bằng cấp

Để xác thực bằng cấp của một sinh viên cần 3 thông tin là id của sinh viên và id của bằng cấp và địa chỉ của contract.

Người muốn xác thực xuất trình 3 thông tin trên, hệ thống sẽ gửi request lấy dữ liệu trên blockchain là các mã hash của bằng cấp và dữ liệu học tập, chương trình đào tạo tương ứng. Từ các mã hash này hệ thống sẽ tự động lấy về các dữ liệu bằng cấp , chương trình đào tạo và dữ liệu học tập của sinh viên tương ứng trên IPFS.

Tiếp đó hệ thống sẽ so khớp dữ liệu học tập và chương trình đào tạo để kiểm tra sinh viên đã hoàn thành chương trình đào tạo chưa.

Cuối cùng thông tin về bằng cấp, chương trình đào tạo , dữ liệu học tập và kiểm tra bằng cấp hợp lệ sẽ được hiển thị về cho người muốn xác thực.

