BỘ GIÁO DỤC VÀ ĐẠO TẠO

TRƯỜNG ĐẠI HỌC PHENIKAA

★ ★ ★

BÁO CÁO ĐỒ ÁN CƠ SỞ

NGHIÊN CỨU VỀ BLOCKCHAIN & SMART CONTRACT VÀ XÂY DỰNG ỨNG DỤNG DEMO

PHENIKAA UNIVERSITY

Giáo viên hướng dẫn: Mai Xuân Tráng

Sinh viên thực hiện: Nguyễn Ngọc Thanh - 20010986
Nguyễn Văn Mạnh Duy - 20010950
Nguyễn Minh Phú - 20010978
Ngô Văn Thuận - 20010988

Hà Nội, tháng 12 năm 2022

MỤC LỤC

LỜI MỞ ĐẦU

Internet xuất hiện không chỉ phục vụ cho việc gửi email hay tải phần mềm mà nó còn là động lực để phát triển kinh tế toàn cầu. Trong thực tế, internet đã trở thành trình điều khiển của nền kinh tế. Sự xuất hiện của Internet và các mạng cục bộ đã giúp cho việc trao đổi thông tin trở nên nhanh chóng, dễ dàng hơn. Email cho phép chúng ta nhận hay gửi thư ngay trên máy tính của mình, Ebusiness cho phép thực hiện giao dịch, buôn bán trên mạng... Cũng giống như Internet, blockchain xuất phát như một trào lưu với đồng tiền ảo Bitcoin.

Sự phát triển của internet cũng đồng hành với những tổn thất sau các cuộc tấn công mạng, gây ảnh hưởng lớn đến nền kinh tế cũng như xã hội. Theo cuộc khảo sát của hãng phân tích Grant Thornton, khoản tiền mà doanh nghiệp mất vào tay tin tặc ở Châu Á-Thái Bình Dương lên tới 81,3 tỉ dollar trong vòng 12 tháng (tính đến cuối tháng 9/2015). Mức tổn thất từ các đợt tấn công mạng ở Châu Á nhiều hơn Bắc Mỹ tới 20 tỉ USD và EU với con số tương tự, chiếm hơn 25% tổng mức tổn thất của thế giới (315 tỉ USD)... Tại Việt Nam cũng xảy ra tình trạng mất an toàn với các tài khoản gửi ngân hàng, điển hình như vụ tấn công vào Vietcombank. Tháng 2/2016, thông tin về việc Ngân hàng Trung Ương Bangladesh bị tin tặc đánh cắp 101 triệu USD gây chấn động thế giới là một bài học cho bất cứ tổ chức nào. Sự cố xảy ra được cho là do Ngân hàng nước này sử dụng bộ định tuyến cũ giá 10 USD mà không có bất cứ một hệ thống tường lửa nào. Số tiền tổn thát trong vụ này có thể lên đến hớn 1 tỷ USD nếu nhưtin tặc không viết sai lỗi chính tả. Từ những rủi ro an ninh mạng nên các tổ chức tài chính cần những công nghệ mới, ví dụ nhưnền tảng của đồng tiền số Bitcoin, chính là Blockchain, được kỳ vọng không chỉ nhằm cắt giảm chi phí ngân hàng mà còn đảm bảo tính an toàn và xa hơn nữa là cách mạng hóa các giải pháp bảo mật.

Nhận thấy Blockchain là một công nghệ rất phù hợp với mức độ bảo mật cao, tính minh bạch cao và không thể chối bỏ lịch sử giao dịch. Nhóm chúng em quyết định nghiên cứu và thực hiện triển khai ứng dụng minh họa để hiểu được công nghệ này cũng như nhằm mục đích xóa bỏ những gian lận trong các giao dịch hiện đại.

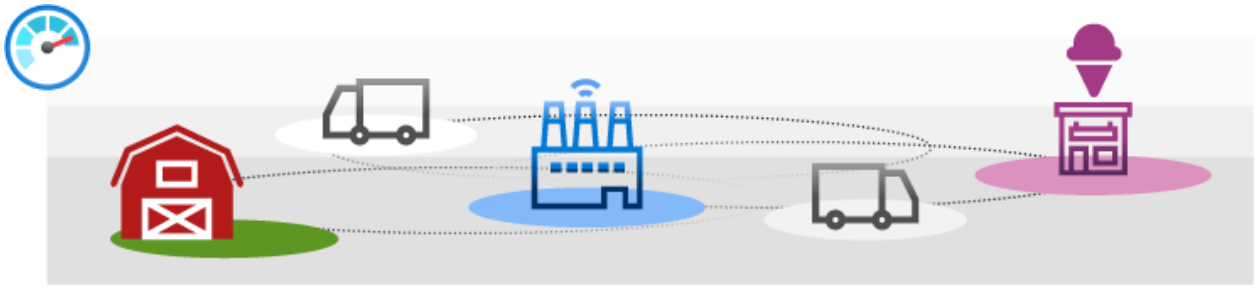
CHƯƠNG 1. TỔNG QUAN VỀ BLOCKCHAIN

1.1. Giới thiệu

Việc triển khai một giải pháp trên nhiều công ty có thể là một thách thức vì bạn cần tin tưởng vào dữ liệu từ các đối tác. Trong hầu hết các trường hợp, chúng ta sử dụng cơ sở dữ liệu trung tâm Central Database. Dữ liệu được lưu trữ ở một vị trí nguồn gốc đáng tin cậy nhất. Công ty duy trì cơ sở dữ liệu phải được tin cậy là cơ quan trung tâm của dữ liệu.

Block Chain cho phép bạn triển khai quy trình kinh doanh khi bạn cần tin tưởng vào dữ liệu và người tham gia mà không cần sử dụng cơ sở dữ liệu trung tâm.

Giả sử bạn là kiến trúc sư tại một công ty kem sữa. Bạn sử dụng một chuỗi cung ứng để nhận các mặt hàng sữa tươi từ nhiều công ty sữa. Công ty của bạn vận chuyển kem đóng gói đến các nhà bán lẻ khác nhau. Đã có vấn đề về chất lượng và an toàn thực phẩm do bảo quản ở nhiệt độ không phù hợp trong quá trình vận chuyển. Bởi vì nhiều công ty chịu trách nhiệm vận chuyển và lưu trữ sản phẩm, nên rất khó để xác định bên có lỗi trong chuỗi cung ứng. Bạn muốn tạo một hệ thống xác định các vấn đề trong chuỗi cung ứng một cách nhanh chóng. Mỗi công ty trong chuỗi cung ứng đều muốn tích hợp các hệ thống hiện có của họ với giải pháp và kiểm tra độc lập các lô hàng nếu có thu hồi về an toàn thực phẩm.



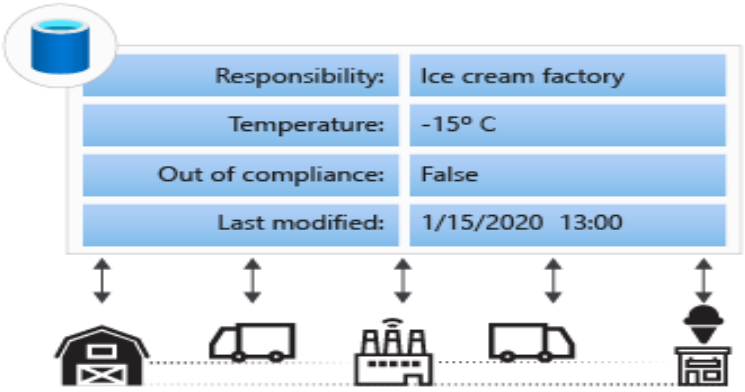
1.2. Blockchain là gì ?

Block Chain là một công nghệ lưu trữ và thực thi sử dụng mật mã để làm cho việc thay đổi lịch sử trước đó trở nên vô cùng khó khăn. Nó cho phép người tham gia chia sẻ luồng công việc bằng cách theo dõi các thay đổi trên sổ cái mà đã được chia sẻ.

Quay trở lại với mô hình công ty kem sữa trên, làm thế nào để chúng ta phát hiện ra vấn đề về chất lượng hoặc an toàn thực phẩm do bảo quản ở nhiệt độ không phù hợp trong quá trình vận chuyển? Chúng ta cần theo dõi bên chịu trách nhiệm và nhiệt độ, đồng thời ghi nhật ký các thay đổi.

1.2.1. Tại sao không sử dụng cơ sở dữ liệu tập trung (*centralized* database)?

Chúng ta có thể sử dụng cơ sở dữ liệu tập trung mà tất cả những người tham gia sử dụng để theo dõi các lô hàng. Trong nhiều trường hợp, cơ sở dữ liệu tập trung là giải pháp phù hợp. Giả sử chúng ta có một cơ sở dữ liệu tập trung lưu trữ thông tin chi tiết về lô hàng và người chịu trách nhiệm hiện tại. Trong kịch bản của trên, chúng ta có thể yêu cầu bên nông trại, người giao hàng, nhà máy và nhà bán lẻ sử dụng cùng một cơ sở dữ liệu tập trung.

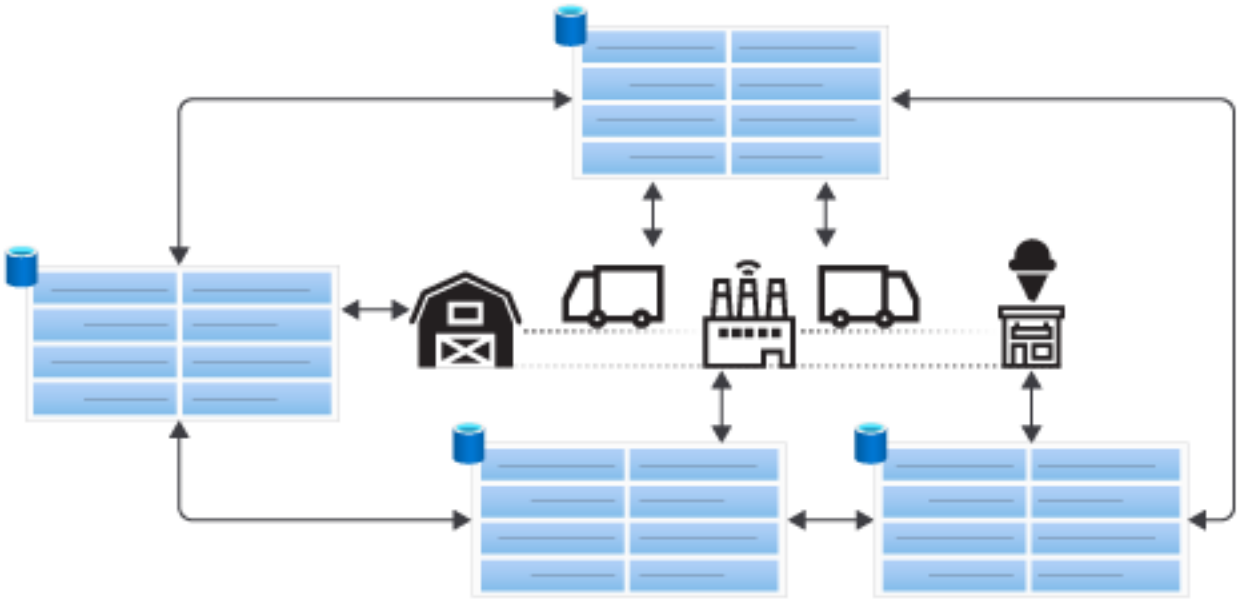


Ưu điểm của cơ sở dữ liệu tập trung là dễ kiểm soát truy cập và nhất quán. Tất cả các bên liên quan đều sử dụng cùng một cơ sở dữ liệu và có một cơ quan đáng tin cậy kiểm soát quyền truy cập. Bởi vì chỉ có một cơ sở dữ liệu, tất cả những người tham gia đang sử dụng cùng một bộ dữ liệu. Tất cả những người tham gia cần tin tưởng rằng cơ sở dữ liệu là chính xác và hơn thế nữa, họ cần tin tưởng chủ sở hữu cơ sở dữ liệu sẽ không sửa đổi lịch sử dữ liệu vì bất kỳ lý do gì.

Điều gì xảy ra nếu kịch bản của chúng tôi không cho phép một cơ sở dữ liệu tập trung đáng tin cậy? Điều gì sẽ xảy ra nếu không có công ty nào muốn chịu trách nhiệm lưu trữ cơ sở dữ liệu tập trung? Có lẽ không thể đáp ứng các yêu cầu về tích hợp hệ thống với tất cả các bên liên quan.

1.2.2. Cơ sở dữ liệu phân tán (*Distributed* database)

Điều gì sẽ xảy ra nếu mỗi người tham gia có thể có bản sao của cơ sở dữ liệu cho riêng họ? Cơ sở dữ liệu phân tán sử dụng nhiều bản sao của cơ sở dữ liệu và các thay đổi được đồng bộ hóa. Trong kịch bản của chúng tôi, chúng tôi có thể yêu cầu nhà cung cấp, người giao hàng, nhà máy và nhà bán lẻ sử dụng cơ sở dữ liệu phân tán của riêng họ.



Mỗi trang trại, nhà máy, người giao hàng và cửa hàng đều sử dụng cơ sở dữ liệu phân tán của riêng họ. Các thay đổi cơ sở dữ liệu được đồng bộ hóa giữa các bản sao.

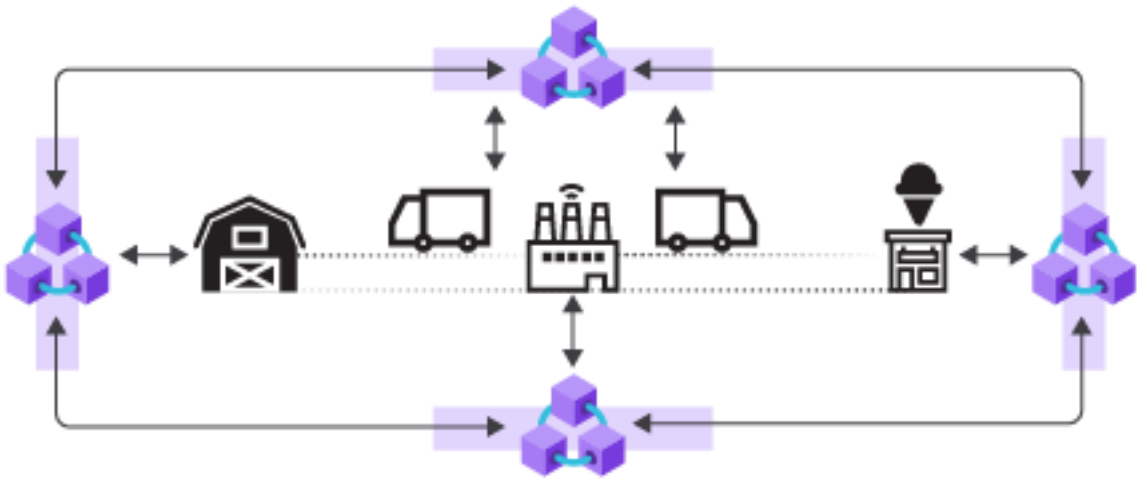
Ưu điểm của cơ sở dữ liệu phân tán là mỗi người tham gia có một bản sao của cơ sở dữ liệu. Trong hầu hết các trường hợp, việc kiểm soát quyền truy cập và tích hợp các hệ thống của bạn và xử lý trong bản sao cơ sở dữ liệu của riêng bạn sẽ dễ dàng hơn. Tuy nhiên, cần phải đồng bộ hóa các thay đổi đối với từng cơ sở dữ liệu. Xử lý các lỗi và xung đột có thể làm tăng thêm các vấn đề phức tạp và làm ảnh hưởng đến sự toàn vẹn dữ liệu.

1.2.3. Sổ cái phân tán (*Distributed* ledger)

Công nghệ Block Chain còn được giới thiệu là sổ cái phân tán. Cũng giống như sổ kế toán, sổ cái phân tán là lịch sử của toàn bộ giao dịch. Mỗi giao dịch trong sổ cái ảnh hưởng đến trạng thái cuối cùng.

Mạng lưới Block Chain được phân phối giữa những người tham gia và các bên liên quan. Mạng liên kết cung cấp cho mỗi đối tác khả năng hiển thị mọi giao dịch xảy ra trên mạng.

Mỗi trang trại, nhà máy, người giao hàng và cửa hàng đều sử dụng sổ cái phân tán của riêng họ. Các giao dịch được gửi đến tất cả các điểm trong mạng.



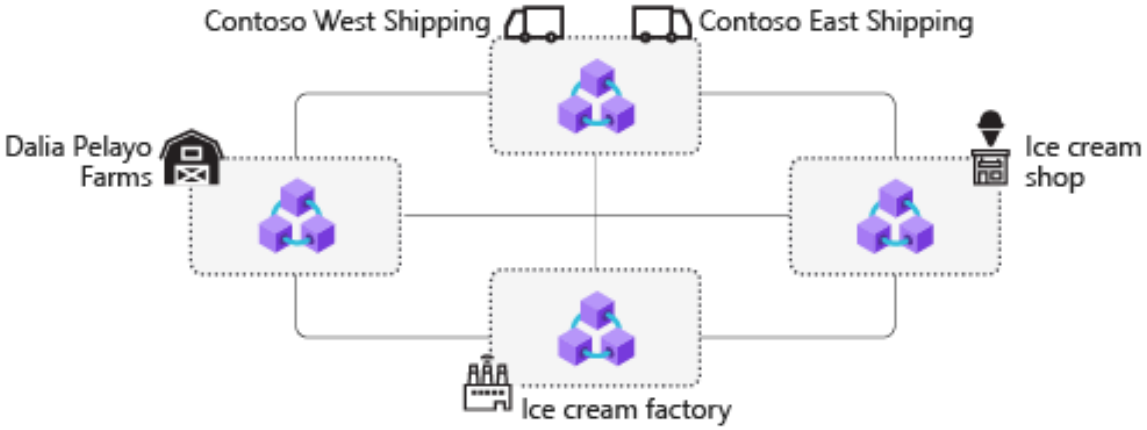
Block Chain sử dụng các quy tắc đồng thuận để đảm bảo dữ liệu nhất quán giữa các nút. Nó cũng sử dụng mật mã để cho phép người tham gia tin tưởng vào dữ liệu. Cụ thể, nó ngăn chặn bất kỳ một cá nhân hoặc một nhóm người dùng sửa đổi lịch sử. Vì blockchain được phân cấp nên các giải pháp có thể sử dụng cơ sở dữ liệu phi tập trung sẽ hoạt động tốt nhất.

1.3. Nguyên lí hoạt động của BlockChain

1.3.1. Dữ liệu được phân phối như thế nào?

Trong kịch bản trên ,có nhiều công ty cùng tham gia. Chúng ta có thể có một cơ sở dữ liệu tập trung tại công ty chế biến sữa. Tuy nhiên, không ai tham gia muốn trở thành cơ quan trung ương. Chúng tôi có thể sử dụng một distributed ledger Block Chain. Sử dụng Block Chain loại bỏ sự cần thiết của một cơ quan trung ương. Ngoài ra, mỗi người tham gia có sử dụng nút Blockchain có một bản sao của sổ cái để họ có thể tự kiểm tra và tích hợp với hệ thống của mình. Tuy nhiên, không có yêu cầu đối với mỗi công ty phải có nút riêng. Các nút có thể được chia sẻ giữa các đối tác.

Mỗi nút được kết nối với các nút khác bằng mạng Block Chain. Ví dụ: *trang trại Dalia Pelayo*, *nhà máy kem* và *cửa hàng kem* đều có một nút để quản lý. *Contoso West* và *Contoso East* là những đối tác riêng biệt có chung công ty mẹ. *Contoso* có một nút. Không cần phải có mối quan hệ 1-1 của các nút với các công ty.

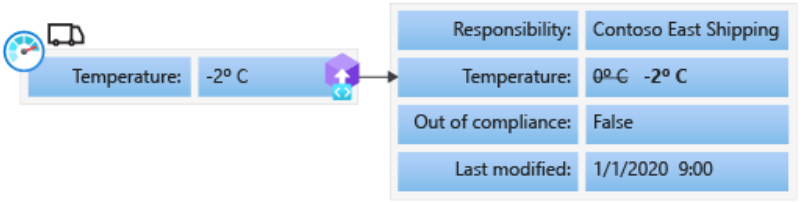


Một số nút blockchain của người tham gia được kết nối bởi một mạng.

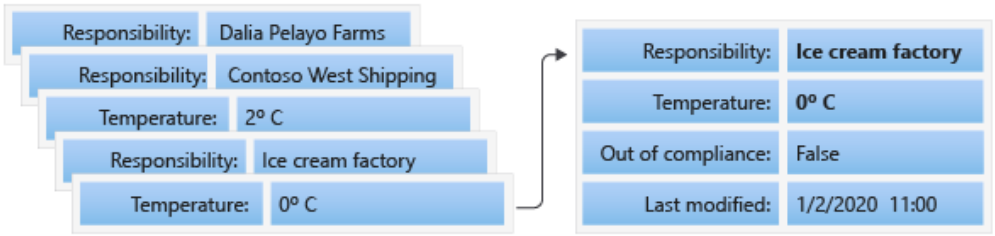
1.3.2. Thay đổi trạng thái

Dữ liệu trong Block Chain đại diện cho trạng thái. Đó là lý do tại sao các digital token như tiền điện tử rất phù hợp với Block Chain. Nếu chúng ta nghĩ về quyền sở hữu tiền tệ vật chất, thì một đồng xu chỉ có thể nằm trong túi của một người tại một thời điểm. Nếu đồng xu nằm trong túi của bạn, trạng thái sở hữu là của bạn. Nếu bạn đưa đồng xu cho bạn của mình, trạng thái sẽ thay đổi thành bạn của bạn sở hữu đồng xu. Trong kịch bản trên, các lô hàng di chuyển qua chuỗi cung ứng. Trách nhiệm đối với sản phẩm phải được chuyển giao. Dữ liệu chúng tôi quan tâm là bên chịu trách nhiệm, nhiệt độ và liệu sản phẩm có tuân thủ hay không.

Block Chain sử dụng các giao dịch để thay đổi trạng thái của dữ liệu từ giá trị này sang giá trị khác. Ví dụ, chúng ta cần biết liệu kem có được bảo quản ở nhiệt độ đủ lạnh hay không. Trong một lô hàng kem, một cảm biến nhiệt độ báo cáo nhiệt độ định kỳ. Nhiệt độ được báo cáo là một giao dịch được gửi đến một nút giao dịch Block Chain.

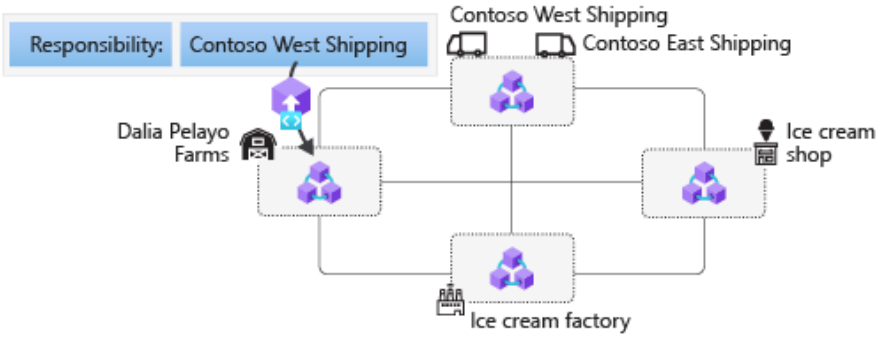


Trong kịch bản kem, khi một lô hàng được gửi qua chuỗi cung ứng, một giao dịch sẽ được gửi mỗi khi trạng thái thay đổi. Ví dụ, hình minh họa cho thấy các giao dịch cho một lô hàng điển hình đến nhà máy kem. Mỗi giao dịch thay đổi bên chịu trách nhiệm hoặc nhiệt độ. Trạng thái hiện tại của sổ là các giao dịch được áp dụng theo thứ tự.

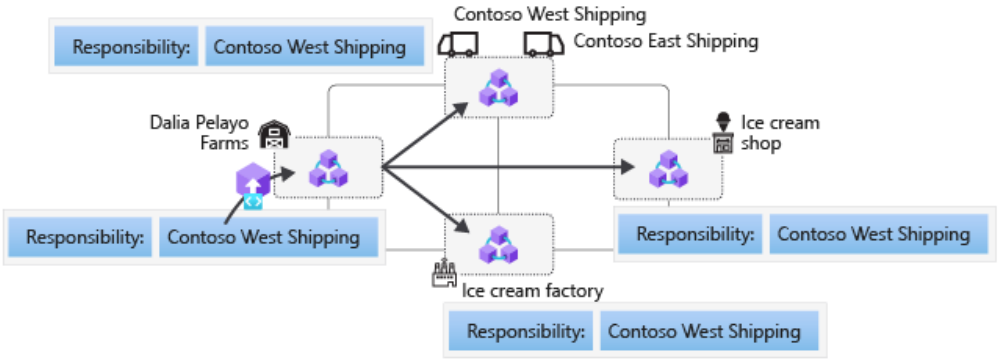


Danh sách các giao dịch theo thứ tự thay đổi bên chịu trách nhiệm và nhiệt độ. Sự kết hợp của các giao dịch dẫn đến trạng thái sổ cái dựa trên thứ tự thay đổi giao dịch.

Khi gửi đi một giao dịch, bạn gửi nó đến một nút giao dịch Block Chain. Giả sử *Dalia Pelayo Farms* gửi một lô hàng sữa bằng *Contoso West Shipping*. Hệ thống giao hàng của *Dalia Pelayo Farms* gửi một giao dịch đến nút Block Chain của họ. Giao dịch cập nhật trách nhiệm vận chuyển từ nông dân đến *Contoso West Shipping*.



Block Chain gửi giao dịch trên toàn mạng Block Chain. Mỗi nút nhận được một bản sao của giao dịch.



Mỗi nút xử lý giao dịch nhưng nó vẫn yêu cầu xác thực bằng cơ chế đồng thuận. Sự đồng thuận cho phép tính nhất quán và tin cậy đối với sổ cái phân tán.

1.3.3. Làm thế nào bạn có thể tin tưởng dữ liệu trong sổ cái là nhất quán?

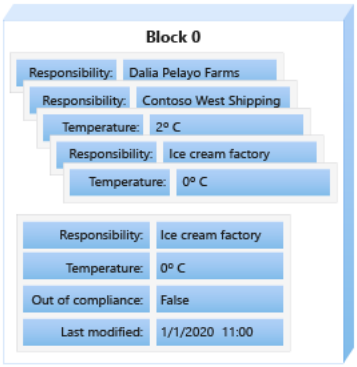
Trong một mạng phân tán, rất khó để xác định điều gì là đúng vì các thay đổi dữ liệu phải xảy ra trên tất cả các nút. Điều gì xảy ra nếu nhà máy sản xuất kem không nhận được giao dịch vận chuyển? Làm sao họ biết nếu sữa đã được vận chuyển từ trang trại? Điều gì sẽ xảy ra nếu xe tải đông lạnh của công ty vận chuyển bị hỏng và sữa bị hỏng? Công ty vận chuyển có thể sửa đổi sổ cái của họ để trốn tránh trách nhiệm không?

Block Chain sử dụng cơ chế đồng thuận để xác thực và đồng ý về dữ liệu trên tất cả các nút Block Chain. Sự đồng thuận cung cấp một cách để tất cả các nút phi tập trung đến cùng một trạng thái. Khi chuyển giao giá trị hoặc trách nhiệm, thứ tự quan trọng. Ví dụ: nếu bạn chuyển quyền sở hữu ô tô cho bạn của mình, thì bạn cũng không thể chuyển quyền sở hữu cho đồng nghiệp tại nơi làm việc. Sự đồng thuận đảm bảo thứ tự đúng của các giao dịch và tính toàn vẹn của Block Chain. Là một phần của sự đồng thuận, một nhóm các giao dịch được xác thực là Block và mạng phải đồng ý nếu khối đó được đưa vào Block Chain.

Có một số thuật toán đồng thuận blockchain bao gồm bằng chứng công việc, bằng chứng cổ phần và bằng chứng về thẩm quyền. Mỗi thuật toán giải quyết tính nhất quán theo một cách khác nhau. Nói một cách đơn giản, sự đồng thuận cung cấp một phương án để sổ cái phân tán đạt đến trạng thái chung.

1.3.4. Block là gì?

Block là một cụm dữ liệu trong Blockchain lưu trữ thông tin giao dịch. Số lượng giao dịch trong Block thường dựa trên thời gian. Ví dụ: hình minh họa cho thấy Block chứa các giao dịch đã xảy ra trong 10 phút qua.



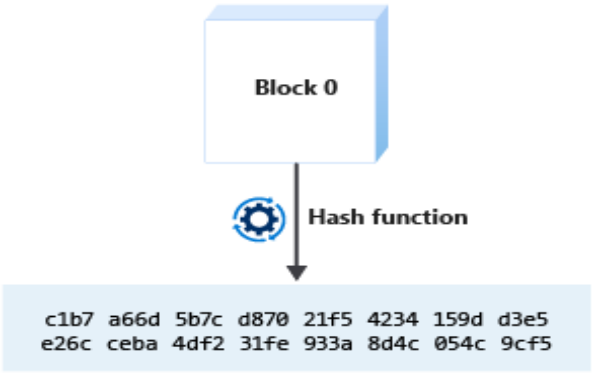
Thông qua sự đồng thuận, các khối được xác thực sẽ được thêm vào blockchain tại mỗi nút. Vì tất cả các nút có cùng block trong chuỗi nên sổ cái sẽ được nhất quán trên toàn mạng lưới. Kết quả là, tất cả các nút đều chứa cùng một dữ liệu đã được xác thực theo thứ tự đã thỏa thuận.

1.3.5. Làm thế nào bạn có thể tin tưởng sổ cái là bất biến?

Bạn nghĩ rằng nếu bạn có quyền kiểm soát sổ cái trong nút của mình, bạn chỉ cần thay đổi dữ liệu trong bản sao của mình. Làm thế nào nó có thể là bất biến?

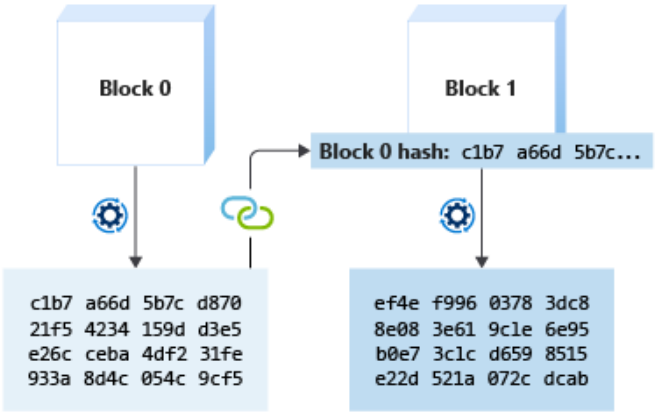
Blockchain sử dụng hàm băm mật mã để tạo liên kết giữa các khối. Bằng cách liên kết các khối, thứ tự của các giao dịch có thể được thống nhất thông qua thuật toán đồng thuận. Băm mật mã là một thuật toán ánh xạ dữ liệu có kích thước tùy ý thành biểu diễn bit có kích thước cố định. Bạn có thể coi nó như một dấu vân tay kỹ thuật số. Bitcoin sử dụng thuật toán băm SHA-256. Nếu bạn đã sử dụng hàm băm SHA-256 trên tài liệu 100 trang, đầu ra của hàm là giá trị băm 256 bit. Nếu bạn chỉ thay đổi một ký tự trong tài liệu và tạo lại hàm băm, thì đầu ra là một giá trị băm 256 bit khác. Bây giờ, hãy

tưởng tượng nếu chúng ta sử dụng Block làm đầu vào cho hàm băm. Đầu ra là một giá trị băm duy nhất cho dữ liệu trong khối.

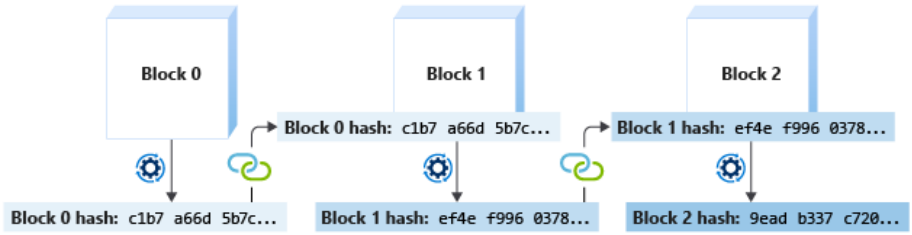


Block được gửi thông qua hàm băm và hàm băm mật mã được tạo.

Blockchain sử dụng hàm băm để phát hiện xem có bất kỳ thay đổi nào được thực hiện đối với các khối hay không. Bằng cách bao gồm giá trị băm của khối trước đó khi tạo hàm băm của khối tiếp theo, các khối được liên kết với nhau thông qua các giá trị băm.



Blockchain cung cấp sự tin tưởng bằng cách sử dụng giá trị băm để chứng minh lịch sử dữ liệu không thay đổi. Bằng cách bao gồm hàm băm của khối trước đó khi tạo khối mới, một chuỗi giao dịch bất biến được tạo theo thứ tự.



Nếu bất kỳ khối nào được sửa đổi trong chuỗi, hàm băm của các khối sau đó sẽ khác. Kết quả là, phát hiện ra sự khác biệt.

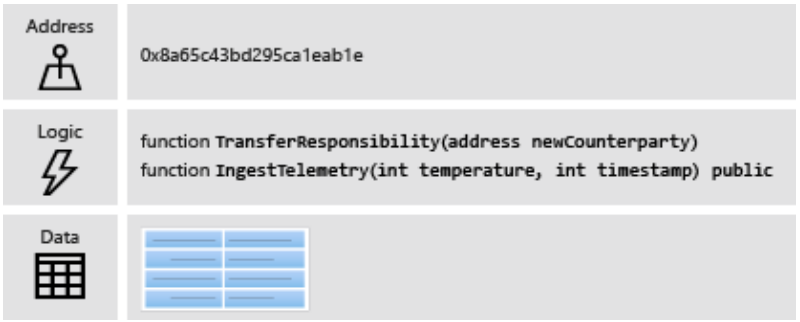
1.3.6. Logic đáng tin cậy

Blockchain cho phép lưu trữ dữ liệu nhất quán và có thể tin cậy được. Làm cách nào để thêm logic mà được thực thi giống nhau tại mỗi nút?

Trong kịch bản này, chúng tôi cần logic để chuyển giao trách nhiệm sản phẩm từ người tham gia này sang người tham gia khác. Chúng tôi cũng cần sử dụng dữ liệu từ cảm biến nhiệt độ IoT để biết liệu nhiệt độ có quá cao hay không.

Ứng dụng phi tập trung (DApp) là một ứng dụng trên hệ thống máy tính phân tán. Trong module này, chúng tôi sẽ tập trung vào việc sử dụng giao thức blockchain Ethereum. Ethereum DApps được gọi là smart contract. Smart contract chứa logic được thực thi như một phần của giao dịch. Trên Ethereum, bạn lập trình logic bằng ngôn ngữ lập trình có tên là Solidity.

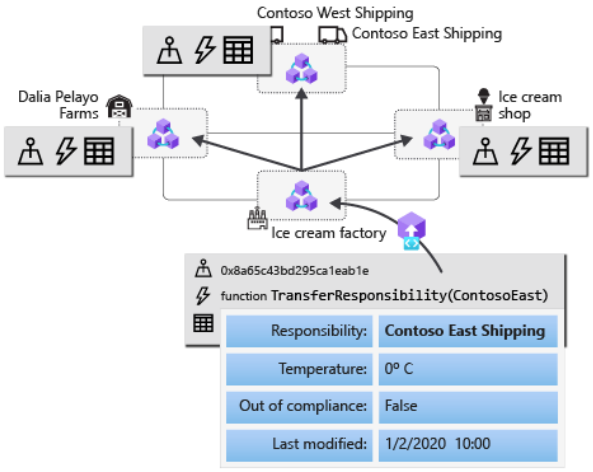
Smart contract được triển khai vào blockchain và được tham chiếu bởi một địa chỉ. Để sử dụng smart contract, bạn tạo một phiên bản. Một phiên bản smart contract chứa dữ liệu trạng thái và logic chương trình. Trong kịch bản này, một phiên bản smart contract chứa dữ liệu như người tham gia chịu trách nhiệm, vị trí và liệu nhiệt độ của sản phẩm có nằm ngoài quy định hay không. Ví dụ, ta có thể thực thi các chức năng để chuyển giao trách nhiệm hoặc nhận phép đo nhiệt độ từ xa.



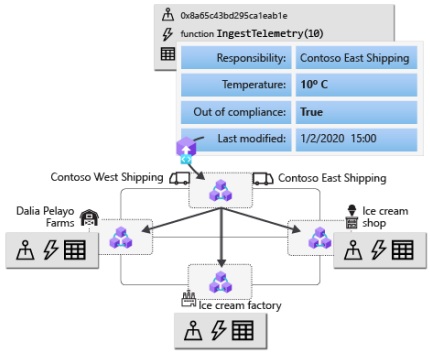
Các thành phần của smart contract bao gồm địa chỉ, logic và dữ liệu.

Khi trách nhiệm của một sản phẩm chuyển giao cho một bên khác, một giao dịch được thực hiện. Smart contract logic cập nhật dữ liệu trạng thái. Trong kịch bản kem, hệ thống vận chuyển của nhà máy sản xuất kem tạo ra

một phiên bản smart contract cho một lô hàng kem mới. Hệ thống vận chuyển của nhà máy gửi một giao dịch gọi hàm TransferResponsibility để chuyển trách nhiệm vận chuyển lô hàng cho công ty vận chuyển Contoso East. Mạng blockchain gửi giao dịch đến tất cả các nút. Logic smart contract thực thi tại mỗi nút.



Điều gì sẽ xảy ra nếu trong quá trình vận chuyển, bộ phận làm lạnh bị hỏng và nhiệt độ của kem vượt quá mức đóng băng? Cảm biến nhiệt độ IoT giám sát nhiệt độ kem và gửi giao dịch định kỳ. Nếu nhiệt độ trên mức đóng băng, logic smart contract sẽ đánh dấu lô hàng là không tuân thủ.



Bởi vì giao dịch được bao gồm trong một chuỗi các khối, nên có một bản ghi bất biến về thời điểm lô hàng trở nên không tuân thủ. Cửa hàng kem có thể từ chối giao hàng và có thể tránh được các vấn đề về an toàn thực phẩm.

Cũng giống như dữ liệu trong blockchain, smart contract là bất biến. Sau khi triển khai, logic không thể thay đổi. Do đó, bạn có thể tin tưởng rằng logic hợp đồng thông minh luôn thực thi giống nhau trên tất cả các nút. Mọi thay đổi về mã đều yêu cầu hợp đồng thông minh mới được triển khai tại một địa chỉ mới.

CHƯƠNG 2. INTERNET COMPUTER (ICP)

2.1. Tổng quan về khái niệm và lịch sử phát triển của internet computer

2.1.1. Khái niệm

Internet Computer là một giao thức lớp 1 được phát triển bởi DFINITY Foundation* được phát triển bằng ngôn ngữ Motoko, nhằm mục đích trở thành một mạng blockchain phát triển internet. Internet Computer dự định mở rộng chức năng của internet công cộng để nó có thể lưu trữ phần mềm phụ trợ trên một mạng phi tập trung, được cung cấp bởi các smart contract và mật mã chuỗi khối (chain key cryptography). Nó sử dụng cryptography để tạo ra một blockchain an toàn hơn và có thể sử dụng được, có thể chạy ở tốc độ web, nghĩa là chúng ta có thể xây dựng các ứng dụng web 100% trên blockchain, và luôn luôn hoạt động.

2.1.2. Lịch sử của Internet computer

- Tháng 10 năm 2016: Tổ chức DFINITY do Dominic Williams bắt đầu, có trụ sở tại Zurich, Thụy Sĩ.

- Tháng 2 năm 2017: Huy động được 4,2 triệu đô la trong một đợt gây quỹ hạt giống (tăng gấp 4 lần so với mục tiêu vốn hóa mềm).

- Tháng 8 năm 2017: Có kế hoạch theo dõi vòng hạt giống ngày 17 tháng 2 với một cuộc gây quỹ chính, nhưng đã quyết định theo một con đường khác.

- Tháng 2 năm 2018: Huy động được 61 triệu đô la từ Andreesen Horowitz và Polychain Capital.

- Tháng 5 năm 2018: Phân phối 35 triệu đô la mã thông báo DFINITY trong một đợt airdrop.

- Tháng 8 năm 2018: Huy động được 102 triệu đô la từ Andreesen Horowitz, Polychain Capital, SV Angel, Aspect Ventures, Village Global, Multicoin Capital, Scalar Capital và Amino Capital, KR1, cũng như các thành viên cộng đồng DFINITY.

- Tháng 11 năm 2019: SDK và Motoko.

- Tháng 1 năm 2020: mạng demo.

- Tháng 6 năm 2020: mở mạng thử nghiệm cho một số nhà phát triển nhất định.

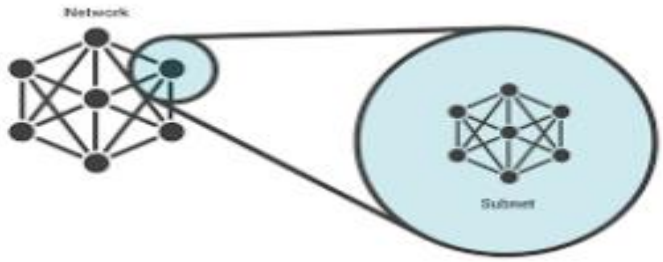
- Tháng 9 năm 2020: Hệ thống thần kinh mạng được công bố / tính kinh tế của hệ sinh thái vi mạch.

- Tháng 12 năm 2020: Ra mắt mạng chính alpha của Internet Computer.

- Tháng 5 năm 2021: Mercury phát hành chính thức: Sự kiện phát sinh máy tính Internet; ra mắt mã thông báo mainnet và ICP.

2.2. Các thành phần của Internet computer

- Subnets (mạng con): là một nhóm các node (nút). Bất kỳ thứ gì chạy trên một mạng con cụ thể đều được sao chép qua một đến nhiều nút trong mạng con. Định nghĩa kỹ thuật hơn (và chính xác hơn) về mạng con là một nhóm phân tán các nút trên phạm vi địa lý và quyền tài phán.



- Nodes (nút): là một máy duy nhất chạy ở một nơi nào đó trên thế giới. Các node kết hợp sức mạnh tính toán của chúng và chạy Internet Computer Protocol.

- Canisters (hộp): là nơi một ứng dụng chạy trên một mạng con (hoặc một nhóm các nút). Điều này có nghĩa là hộp là mã, trạng thái, bộ nhớ và lưu trữ cho các ứng dụng Internet Computer.

- The Nervous Network System (NNS): Đây là một hệ thống biểu quyết các tế bào thần kinh (neutron). Bất kỳ ai cũng có thể mua ICP trên một sàn giao dịch và đặt nó vào một tế bào thần kinh để có thể gửi đề xuất thay đổi giao thức hoặc bỏ phiếu cho các đề xuất mà người khác đã gửi. Hệ thống biểu quyết các nơ-ron này là thứ chi Internet Computer. (Có thể hiểu bạn mua 1 ghế trong hội đồng, số neutron bạn càng cao thì quyền biểu quyết của bạn càng lớn).

- **Neuron** (tế bào thần kinh): Một neuron có thể chứa ICP đặt cọc (stake). Khi ICP được đặt cược và độ trễ giải thể (mở khóa unlock) lớn hơn 6 tháng, nơ-ron là một thực thể biểu quyết trong NNS.

- **Proposal** (đề xuất): Để thay đổi Internet Computer theo bất kỳ cách nào, bạn phải thực hiện thông qua đề xuất với NNS. Đề xuất sẽ phác thảo một thay đổi cụ thể và tất cả mọi người có ICP đặt cọc trong tế bào thần kinh sẽ bỏ phiếu cho nó để quyết định có áp dụng thay đổi hay không.

- **Votes** (quyền biểu quyết): Mỗi nơ-ron nhận được quyền biểu quyết dựa trên số lượng ICP được đặt trong nó, thời gian ICP đã được đặt trong nơ-ron và thời gian ICP phải bị khóa (còn gọi là độ trễ hòa tan). Khi một nơ-ron bỏ phiếu cho một đề xuất NNS, sẽ kiếm được phần thưởng.

- **ICP**: Internet Computer Protocol, ICP, là một mã thông báo tiện ích có thể được mua tại các sàn giao dịch thông thường (Coinbase, Binance) và được đặt trong một nơ-ron để tham gia vào việc quản lý Internet Computer. Bạn cũng nhận được phần thưởng trong ICP cho việc đặt cược và bỏ phiếu cho các đề xuất NNS.

- **Cycles**: ICP có thể được chuyển đổi thành các Cycles (chu kỳ). 1 nghìn tỷ chu kỳ giống như 1 SDR (tiền tệ quốc tế). Tỷ lệ chuyển đổi giữa các chu kỳ và ICP luôn thay đổi dựa trên một đề xuất NNS xảy ra sau mỗi 10 phút. Chu kỳ là một hoạt động được thực hiện bởi máy tính, vì vậy chu kỳ là một dạng chi tiết hơn của tính toán chi phí. Các hộp (xem định nghĩa ở trên) thực hiện các phép tính trên Internet Computer và sử dụng các chu kỳ để thực hiện các phép tính đó.

2.3. Lợi ích chính của ICP

Chúng ta đã và đang xây dựng các ứng dụng trên internet trong nhiều thập kỉ tuy nhiên có một số vấn đề lớn đối với các ứng dụng web tập trung và ICP có các giải pháp thuyết phục để giải quyết các vấn đề này và sẽ được giải thích rõ hơn ở phần dưới đây:

- Quyền sở hữu cộng đồng.

- Cải thiện bảo mật.

- Phân cấp hoàn chỉnh.

- Dễ sử dụng hơn cho nhà phát triển.

2.3.1. Quyền sở hữu cộng đồng

Khi một công ty công nghệ ra mắt công chúng hoặc được mua lại (các sự kiện thanh khoản lớn đối với một công ty), những người chiến thắng đều ở trong công ty, ở cấp cao nhất: người sáng lập, giám đốc điều hành và nhà đầu tư. Một số nhân viên ban đầu cũng có thể giành chiến thắng và những nhân viên còn lại có thể thắng ở một mức độ nào đó. Nhưng người dùng ứng dụng ban đầu không nhận được gì. Người dùng nhóm tiêu điểm không nhận được gì. Người dùng tận tâm hơn 10 năm không nhận được gì.

Sự giàu có được tạo ra từ một cộng đồng được xây dựng dựa trên công nghệ chỉ dành riêng cho những người ở cấp cao nhất.

Đây là một vấn đề có thể sẽ chỉ gia tăng trong những năm tới khi công nghệ tiếp tục phát triển và quyền kiểm soát, quyền lực và ảnh hưởng sẽ tiếp tục được tập trung vào tay những người đã nắm quyền tại các công ty công nghệ.

Giải pháp cho vấn đề này là phân phối của cải cho toàn bộ cộng đồng. Điều này thường xảy ra theo hai cách:

- Airdrop cho người sử dụng ứng dụng

- Bán mã thông báo công khai giai đoạn đầu:

Việc bán mã thông báo giai đoạn đầu này cho phép bất kỳ ai đầu tư vào các công ty yêu thích của họ. Nó hoạt động tốt nhất với các công ty hoàn toàn phi tập trung vì các lý do pháp lý, nhưng phân tán sự giàu có trên một hệ sinh thái tốt hơn nhiều so với công ty tư nhân tiêu chuẩn theo lộ trình IPO hoặc mua lại.

Cả hai chiến lược lan truyền sự giàu có này đều được thực hiện dễ dàng trên Internet Computer. Airdrop phổ biến trong các ứng dụng do tính chất API mở và có thể tương tác của NFT và các tiêu chuẩn mã thông báo trên IC. Và Service Nervous System (SNS) sẽ ra mắt vào năm 2022, sẽ là công cụ bán mã thông báo DAO + chìa khóa trao tay cho các ứng dụng web. Điều này sẽ cho phép các thành viên cộng đồng đầu tư sớm vào các công ty yêu thích của họ để chia sẻ lợi thế khi công ty phát triển và thành công hơn.

2.3.2. Cải thiện bảo mật

- Misconfigurations (Cấu hình sai):

Cấu hình sai là khi máy chủ và / hoặc cơ sở dữ liệu được định cấu hình không đúng cách cho phép truy cập không đúng vào dữ liệu công khai và riêng tư. Đáng buồn thay, điều này xảy ra mọi lúc do sự phức tạp của cơ sở hạ tầng internet. Điều này có thể vô hiệu hóa xác thực đa yếu tố cho quản trị viên, khiến dữ liệu không được mã hóa ở trạng thái nghỉ hoặc không cấp quyền truy cập tài nguyên đúng cách.

Ví dụ, các doanh nghiệp kinh doanh có 2.269 sự cố cấu hình sai mỗi tháng và các doanh nghiệp thường đánh giá thấp số lượng cấu hình sai mà họ gặp phải theo hệ số 10.

Đối với các nhà phát triển xây dựng ứng dụng, có nhiều phần chuyển động trong một thiết lập AWS phức tạp hơn so với khi so sánh với Internet Computer. Vì Internet Computer có ít bộ phận chuyển động hơn, nên việc cấu hình đúng và chính xác sẽ dễ dàng hơn. Bạn chỉ cần chọn một mạng con để triển khai và bạn đã hoàn tất! Thiết lập đơn giản hơn nhiều này dẫn đến ít lỗ hổng định cấu hình sai hơn.

- Phishing (lừa đảo):

Phishing là một cuộc tấn công mạng trong đó ai đó cố gắng yêu cầu bạn cung cấp cho họ thông tin cá nhân của bạn (mật khẩu, khóa cá nhân, v.v.) bằng cách giả vờ là người mà bạn tin tưởng. Phần lớn các cuộc tấn công lừa đảo xảy ra thông qua email. Lừa đảo là loại tội phạm mạng phổ biến nhất và 96% các cuộc tấn công lừa đảo đến qua email.

Phần lớn các ứng dụng trên Internet Computer đang sử dụng Internet Identity để quản lý xác thực. Internet Identity sử dụng sinh trắc học và khóa vật lý (như Yubikey) để xác thực. Điều này có nghĩa là ngay cả khi ai đó muốn đánh cắp mật khẩu của bạn, họ cũng không thể vì Internet Identity không sử dụng mật khẩu và Internet Computer sẽ chỉ lưu trữ thông tin mật mã công khai từ đăng nhập sinh trắc học của bạn. Điều này làm cho thông tin xác thực trên Internet Computer an toàn hơn nhiều so với một nhà cung cấp đám mây truyền thống.

- Security Fixes (Các bản sửa lỗi bảo mật) không được triển khai:

Trong nhiều trường hợp, một lỗ hổng bảo mật được phát hiện và vá (trong kho mã chính), nhưng các bản vá không được áp dụng đồng nhất trên tất cả các trang web và ứng dụng. Trên thực tế, 60% các vụ vi phạm bảo mật đã xảy ra khi một bản sửa lỗi bảo mật đã có sẵn nhưng chưa được thực hiện.

Internet Computer chịu sự điều chỉnh của Service Nervous System (NNS). Khi cần cập nhật, ai đó có thể gửi đề xuất tới NNS, mọi người có thể bỏ phiếu chấp nhận hoặc từ chối đề xuất và sau đó các bản sửa lỗi sẽ tự động được áp dụng trên tất cả các nút trong mạng. Điều này có nghĩa là khi một lỗ hổng bảo mật được phát hiện và khắc phục, bản sửa lỗi có thể được đẩy ra 100% các nút, giảm thiểu hiệu quả lỗ hổng bảo mật này trên Internet Computer.

- Downtime (Thời gian ngừng hoạt động)

Đôi khi, một mạng phân phối nội dung tập trung (CDN) hoặc nhà cung cấp dịch vụ lưu trữ sẽ gặp sự cố và gỡ bỏ một phần mạng internet với nó. Ví dụ:

– Hai lục địa bị ảnh hưởng bởi Dyn Cyberattack vào năm 2016, ảnh hưởng đến Airbnb, Amazon, BBC, CNN, eBay, Netflix và Twitter

– Toàn bộ Đội bay của British Airways ngừng hoạt động vì sự cố CNTT vào năm 2017 – Microsoft Azure đã ngừng hoạt động trong 11 giờ do lỗi của con người trong năm 2018

– Facebook ngừng hoạt động trong 36 giờ vào ngày 13 tháng 3 năm 2019, ảnh hưởng đến 7,5 triệu người

– Máy chủ của Google gặp sự cố vào tháng 12 năm 2020, ảnh hưởng đến YouTube, Google Drive, Google Meet và Gmail

– Lỗi cấu hình nhanh vào ngày 8 tháng 6 năm 2021 đã ảnh hưởng đến Amazon, Reddit, Twitch, Github, Shopify và Spotify.

Trên máy tính kết nối Internet, hệ số sao chép mặc định cho các nút trong mạng con là 13, có nghĩa là ứng dụng của bạn sẽ chạy đồng thời ở 13 nơi khác nhau trên thế giới và sẽ chỉ bị gỡ xuống nếu nhiều trung tâm dữ liệu cùng hoạt động. thời gian. Internet Computer có chức năng này theo mặc định , vì vậy sẽ có ít lần ngừng hoạt động hơn nếu mọi ứng dụng được lưu trữ trên Internet Computer.

2.3.3. Decentralization Benefits (Lợi ích phân quyền) của Internet Computer

Vào ngày 21 tháng 1 năm 2021, một trang web truyền thông xã hội gây tranh cãi có tên Parler đã bị AWS và các nhà cung cấp dịch vụ lưu trữ khác đột ngột gỡ xuống. Cho dù bạn có đồng ý với lý do Parler bị xóa hay không, thì có vẻ gì là lạ khi một công ty công nghệ khổng lồ hoạt động vì lợi nhuận có thể quyết định ai bị cấm và ai không? Không nên để những quyết định này cho cộng đồng?

Vào ngày 8 tháng 1 năm 2021, Tổng thống Hoa Kỳ, Donald Trump, đã bị cấm trên nhiều trang mạng xã hội. Cho dù bạn có đồng ý với lý do anh ta bị cấm hay không, không có gì lạ khi một công ty công nghệ vì lợi nhuận khổng lồ đưa ra quyết định ai bị cấm khỏi nền tảng của họ và ai không? Không nên để những quyết định này cho cộng đồng?

Với các ứng dụng truyền thông xã hội phi tập trung và các ứng dụng khác chạy trên Máy tính Internet, cộng đồng có thể đưa ra các quyết định gỡ xuống này. Quản trị máy tính Internet diễn ra thông qua Service Nervous System (NNS). NNS nhận được các đề xuất và bất kỳ ai đã đặt ICP của họ trong một nơ-ron (với thời gian trì hoãn giải thể ít nhất 6 tháng) đều có thể tham gia bỏ phiếu cho các đề xuất.

Các đề xuất này có thể bao gồm việc cấm một người dùng cụ thể phù hợp với các tiêu chí cụ thể hoặc chúng có thể bao gồm các nguyên tắc hoặc hạn chế về loại nội dung được phép trong một nền tảng cụ thể.

Sự phân quyền của Internet Computer có nghĩa là bất kỳ ai muốn bỏ phiếu để xác định điều gì xảy ra trên Internet Computer đều có thể có quyền đó, miễn là họ phải trải qua quy trình mua và đặt cược ICP thích hợp.

2.3.4. Dễ sử dụng cho nhà phát triển

Các nhà phát triển vi mạch tin rằng việc xây dựng ứng dụng trên Internet Computer đơn giản hơn nhiều so với các dịch vụ tập trung khác.

Thiết lập mọi thứ trên AWS, bạn phải lo lắng về việc tạo đám mây riêng ảo (VPC), tạo mạng con của riêng bạn trong VPC, định cấu hình địa chỉ ip của mạng con của bạn, thiết lập nhóm bảo mật để đảm bảo mọi thứ đều có quyền truy cập và vai trò thích hợp trong mỗi mạng con tương ứng , định cấu hình tường lửa, đảm bảo cấu hình cổng của bạn có ý nghĩa, triển khai trình

cân bằng tải, đảm bảo bạn có chứng chỉ SSL cho tất cả các miền và đảm bảo DNS được thiết lập đúng cách.

Việc triển khai với Internet Computer có thể đơn giản như việc chọn mạng con, hệ số sao chép và cài đặt quyền riêng tư của bạn. Máy tính Internet loại bỏ phần lớn sự phức tạp.

2.4. So sánh Internet computer với các nền tảng blockchain layer 1 khác

Sự đổi mới chính đằng sau Internet Computer là Công nghệ Chìa khóa Chuỗi (Chain Key Technology), với nhiều công nghệ mới được giới thiệu bao gồm: Cơ chế đồng thuận, tạo khóa phân tách không tương tác – Non-Interactive Distributed Key Generation (NI-DKG), hệ thống thần kinh mạng – Network Nervous System (NNS), Internet Identity, v.v.

Trong khi các blockchain khác chỉ đang cố gắng thay đổi nhằm cải thiện một số hạn chế mà Ethereum đang phải đối mặt như tốc độ giao dịch hay phí gas thì Internet Computer còn làm được nhiều hơn thế.

Chúng ta hãy bắt đầu với những yếu tố cơ bản nhất của một nền tảng blockchain:

2.4.1. Cơ chế đồng thuận

Mục đích của cơ chế đồng thuận là xác minh rằng thông tin được thêm vào sổ cái là hợp lệ. Điều này đảm bảo rằng khối sau thêm vào được thể hiện một cách chính xác và tất cả các giao dịch trong mạng đều được cập nhật. Điều này ngăn việc lập lại thông tin nhiều lần hoặc dữ liệu không hợp lệ.

Proof-of-Work (PoW), giao thức đồng thuận phổ biến nhất trong tiền điện tử, lần đầu tiên ra mắt cùng với sự phát minh ra Bitcoin. Ethereum đã áp dụng cơ chế tương tự, hiện nay đã được nâng cấp lên Proof-of-Stake (PoS).

Nhiều blockchain khác đã sao chép mã nguồn của Bitcoin do đó cũng sử dụng mô hình Proof-of-Work.

Mặc dù Proof of Work là một phát minh đáng kinh ngạc, nhưng nó không hoàn hảo. Nó không chỉ cần một lượng điện đáng kể mà còn rất hạn chế về số lượng giao dịch mà nó có thể xử lý cùng một lúc.

Proof-of-Stake (PoS) được tạo ra để thay thế cho Proof-of-Work và giải quyết các vấn đề còn hạn chế của Proof-of-Work. Những ưu điểm chính của Proof-of-Stake là nó làm giảm năng lượng tiêu thụ và cải thiện tốc độ tạo mỗi khối được thực hiện trong vài giây. (mili giây trong trường hợp của Solana, nhưng vẫn chậm hơn 10 lần so với tốc độ của Internet Computer).

Solana, Binance Smart Chain và Avalanche sử dụng cơ chế đồng thuận Proof of Stake. Các blockchain khác sử dụng các thuật toán đồng thuận dựa trên Proof of Stake như:

– Polkadot (Nominated Proof of Stake, NPoS).

– Cardano (Ouroboros).

– Algorand (Pure Proof of Stake, PPoS).

– Zilliqa thì sử dụng thuật toán đồng thuận (PBFT) kết hợp với Proof-of-Work.

Internet Computer sử dụng cơ chế đồng thuận Threshold Relay , một phiên bản được tối ưu hóa của mô hình Proof-of-Stake (PoS). Nó tập trung vào việc tối ưu thời gian xác thực giao dịch bằng cách kết hợp Threshold Relay với các chuỗi chữ ký BLS để giải quyết các vấn đề liên quan cơ chế đồng thuận PoS.

Trong phiên bản hiện tại của Internet Computer, các node tạo ra một số ngẫu nhiên, được gọi là "random beacon", được sử dụng để chọn nhóm nodes tiếp theo và điều khiển các giao thức của nền tảng.

2.4.2. Tốc độ

Khi so sánh giữa các nền tảng blockchain, thông số được những người dùng quan tâm nhất là những thông số liên quan đến tốc độ. Không giống như bảo mật hoặc khả năng mở rộng, tốc độ có các thông số có thể đo lường được giúp xếp hạng người chiến thắng dễ dàng hơn.

Tốc độ giao dịch được tính bằng ba số liệu:

– Kích thước khối: Lượng dữ liệu (tính bằng byte) có thể được chứa trong một khối duy nhất.

– Thời gian tạo khối: Thời gian cần thiết để tạo khối tiếp theo trong chuỗi khối.

– Kích thước giao dịch trung bình: Kích thước giao dịch trung bình trên mạng blockchain.

Thông thường, việc thực hiện phép tính này rất phức tạp bởi thực tế là một số blockchain như Ethereum, đã và đang tăng dần kích thước khối trong những năm qua để đáp ứng nhu cầu giao dịch.

Tốc độ của mạng blockchain ảnh hưởng trực tiếp đến thời gian người dùng cuối thực hiện giao dịch từ tài khoản này sang tài khoản khác. Thời gian này được đo bằng tham số "thời gian xác thực", cho biết khoảng thời gian chúng ta phải đợi để đảm bảo rằng các giao dịch tiền điện tử không thể bị thay đổi, đảo ngược hoặc hủy bỏ sau khi chúng hoàn tất.

Một số blockchain thường sử dụng "Thời gian tạo khối" thay vì sử dụng "Thời gian xác thực" để quảng cáo về tốc độ blockchain của mình. Nhưng họ lại không đề cập đến các thông số như độ trễ (thời gian cần để mạng blockchain xác nhận một giao dịch), được bao gồm trong "thời gian xác thực". Tốc độ thực tế của một blockchain được kiểm tra tại đây.

Chỉ số quan trọng tiếp theo mà chúng ta cần quan tâm đó là là số giao dịch thực hiện trong mỗi giây (TPS).

Giao dịch mỗi giây là số lượng giao dịch mà mạng có khả năng xử lý mỗi giây. Đây chỉ là một con số lý thuyết được tính toán như các giao dịch trên mỗi khối, chia cho thời gian tạo khối.

Solana đã tích cực quảng bá bản thân về khối lượng giao dịch cao và thời gian tạo khối thấp.

Thời gian tạo khối của Solana thực sự nhanh (nó là tốt nhất sau Internet Computer), nhưng nó rất khác thời gian xác nhận. Nó thường sẽ tốn vài khối trước khi có thể thực hiện giao dịch bao gồm cả việc tạo khối cũng như đạt được trạng thái đồng thuận. Solana sử dụng "Optimistic Confirmation" cần 32 phiếu; do đó "Thời gian xác nhận" giao dịch là khoảng 13 giây.

Mặc dù cả hai đều không có thời gian tạo khối tốt bằng Solana, nhưng Algorand và Avalanche đã và đang cải thiện thời gian xác thực giao dịch. Do đó, chúng ta có thể khẳng định rằng, sau Internet Computer, blockchain có tốc độ dữ liệu tốt nhất là Avalanche.

Internet Computer sử dụng công nghệ Keychain của mình để hoàn thành giao dịch giữa các hợp đồng thông minh trong 1-2 giây. Đó là khoảng thời gian người dùng có thể chấp nhận được.

Các ứng dụng cụ thể như trò chơi trực tuyến yêu cầu phản hồi được cung cấp cho người dùng tính bằng mili giây.

Internet Computer giải quyết vấn đề này bằng cách chia việc thực thi chức năng hợp đồng thông minh thành hai loại, được gọi là "lệnh gọi cập nhật" và "lệnh gọi truy vấn".

Sự khác biệt cơ bản của hai lệnh là khi bạn gọi một hàm dưới dạng lệnh gọi cập nhật, bất kỳ thay đổi nào mà nó thực hiện đối với dữ liệu trong bộ nhớ của canister sẽ vẫn tồn tại, trong khi nếu một hàm được gọi dưới dạng lệnh gọi truy vấn, thì bất kỳ thay đổi nào mà nó thực hiện đối với bộ nhớ sẽ bị loại bỏ sau khi hoàn tất truy vấn.

Các lệnh gọi cập nhật tạo ra các thay đổi liên tục và chúng cũng chống được giả mạo vì các giao thức của ICP chạy chúng trên mọi node trên subnet. Tại thời điểm bài viết, thời gian thực hiện lệnh gọi cập nhật: 2s.

Mặt khác, các lệnh gọi truy vấn không tạo thay đổi liên tục. Mọi thay đổi mà chúng thực hiện đối với bộ nhớ sẽ bị loại bỏ sau khi lệnh được thực hiện. Chúng rất hiệu quả và không tốn kém, thời gian thực hiện một lệnh tại thời điểm bài viết là 0.2s. Do chúng không chạy trên tất cả các node trên subnet, điều này cũng có nghĩa là chúng cung cấp mức độ bảo mật thấp hơn.

Trong giai đoạn Genesis, mạng con hệ thống thần kinh mạng NNS đã được khởi chạy với 28 node và mỗi mạng con ứng dụng có 7 node. Kích thước của mạng con được quyết định bằng phiếu bầu của người dùng stake ICP trên NNS. Trong máy tính Internet, mạng con là các blockchains mà mật mã Chain Key kết hợp thành một blockchain duy nhất.

Internet Computer đang liên tục phát triển theo cấp số nhân, với 4.300 node dự kiến vào cuối năm nay, do đó, giao dịch mỗi giây (TPS) của một mạng con sẽ được nhân với số mạng con được tạo ra. Vì thế IC không có giới hạn về TPS.

2.4.3. Khả năng mở rộng

Khả năng mở rộng của mạng blockchain là khả năng hỗ trợ một lượng giao dịch cao và tăng trưởng trong tương lai. Điều này có nghĩa là khi lượng

người dùng và dapps ngày càng tăng thì hiệu suất của blockchain phải không bị ảnh hưởng.

Bitcoin và Ethereum đang bị ảnh hưởng bởi các vấn đề về mở rộng.

Bitcoin và Ethereum đã gặp phải các vấn đề về khả năng mở rộng trong vài năm qua do giới hạn của cơ chế đồng thuận Proof of Work. Hiện tại, Ethereum có thể tận dụng các giải pháp layer 2 để khắc phục các vấn đề về khả năng mở rộng, nhưng các node chạy trên các nền tảng đám mây của các công ty công nghệ lớn như Amazon Web Services (AWS), nó mất đi tính phi tập trung cần có.

Ethereum cũng có kế hoạch chuyển từ Proof of Work sang Proof of Stake trong bản cập nhật có tên là "London". Bản cập nhật này sẽ cho phép nó cải thiện năng lực và khả năng mở rộng của Ethereum lên tới 64 shard chains.

Các shard chain này sẽ cung cấp cho Ethereum nhiều dung lượng hơn để lưu trữ và truy cập dữ liệu, nhưng chúng sẽ không được sử dụng để thực thi code.

Tương tự Ethereum 2.0, với Polkadot là Relay chain, với một số shard được gọi là Parachain. Số lượng Parachain có giới hạn và hiện được ước tính vào khoảng 100. Như tôi đã đề cập trong phần trước, subnet của IC là các blockchains mà công nghệ Chain Key kết hợp thành một blockchain duy nhất để tăng dung lượng của nó theo nhu cầu (dung lượng không giới hạn) và cung cấp một lộ trình để có khả năng mở rộng vô hạn. Số lượng subnet của IC là không giới hạn.

Để đạt được khả năng mở rộng, Binance Smart Chain hy sinh khả năng phi tập trung.

Trong cơ chế đồng thuận của BSC, nó chỉ sử dụng 21 trình xác nhận (Proof of Authority), điều đó làm cho BSC trở thành blockchain thiếu phi tập trung nhất. Trong khi đó, Cardano vẫn đang chờ Hydra, giải pháp layer 2 của họ, giải pháp tương tự mà Matic (Polygon) đã cung cấp Ethereum từ khá lâu rồi.

Tiếp theo là Solana

Giống như cách Bitcoin và Ethereum hy sinh khả năng mở rộng, Solana đã hy sinh khả năng phi tập trung của mình. Bằng chứng lịch sử (PoH) "sáng tạo" ra thêm một vấn đề mới không tồn tại trong các blockchain khác. Mỗi

ngày, giao thức này tạo ra một lượng lớn dữ liệu lịch sử giao dịch cần được lưu trữ (hơn 2 TeraByte mỗi năm).

Kích thước của nó thậm chí còn lớn hơn tổng dữ liệu được tích lũy bởi mười mạng blockchain hàng đầu cộng lại. Solana lưu trữ lượng dữ liệu khổng lồ này trong Arweave (một mạng lưu trữ phi tập trung) do đó các trình xác thực của nó chỉ có thể lưu trữ dữ liệu trong hai ngày.

Bằng cách này, Solana đặt thông tin lịch sử giao dịch của khách hàng vào tay các chain khác do cộng đồng khác quản lý.

Ngoài ra, khả năng mở rộng của Solana đang gặp phải vấn đề gần đây. Solana đã 'sập' sau khi mạng lưới blockchain bị ngừng hoạt động trong hơn 17 giờ. Mạng không thể xử lý sự gia tăng hoạt động, cái mà Solana gọi là "cạn kiệt tài nguyên". Và đây không phải là lần đầu tiên nó gặp phải tình trạng ngừng hoạt động blockchain trước đó đã từng có lần ngừng hoạt động trong khoảng sáu giờ vào tháng 12 năm 2020.

Cuối cùng, tôi sẽ tập trung vào Avalanche và Algorand

Mạng Avalanche là một nền tảng được xây dựng bởi ba chuỗi khối tương thích: Chuỗi giao dịch (X-Chain), Chuỗi nền tảng (P-Chain) và Chuỗi hợp đồng (C-Chain). Các subnet được quản lý trên P-Chain, hoạt động như một mạng lưới mini và tất cả các mạng lưới mini này tham gia để tạo thành mạng Avalanche lớn hơn. Do đó, khả năng mở rộng sẽ phụ thuộc vào số lượng mạng con.

Nhược điểm của Avalanche (và Algorand) là không cung cấp dịch vụ lưu trữ dữ liệu của riêng. Algorand sử dụng mạng lưới chuyển phát nội dung hoàn toàn phi tập trung (IPFS) và Avalanche sử dụng cả Arweave (thông qua mạng Kyve) và Ceramic. Họ sử dụng các dịch vụ phi tập trung này để chia sẻ tệp và lưu trữ dữ liệu.

Tất cả dữ liệu đều được lưu trự on-chain trên IC, đây là một lợi thế quan trọng khác của khả năng mở rộng.

2.4.4. Phí giao dịch

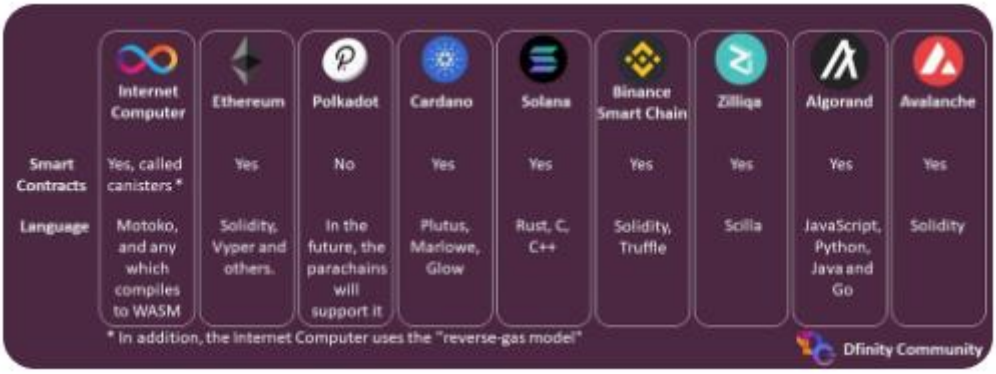
Phí giao dịch thưởng cho người khai thác (PoW) hoặc người xác thực (PoS), những người giúp xác nhận giao dịch.

Trong khi phí Bitcoin phụ thuộc vào quy mô của giao dịch tính bằng byte thì phí giao dịch Ethereum tính đến khả năng tính toán cần thiết để xử lý một giao dịch, được gọi là gas, cũng có giá biến đổi được đo bằng ETH và có liên quan trực tiếp đến lưu lượng truy cập mạng.

Đối với BSC, phí giao dịch là tương tự như một cách làm của Ethereum. Không có gì lạ bởi theo quan điểm của tôi, BSC là một bản sao của Ethereum. Họ đã thay đổi mô hình đồng thuận để cải thiện một số hạn chế của mô hình sau (và cũng phải nói rằng, làm những hạn chế trên ETH trở nên tồi tệ hơn như tính phi tập trung).

Cuối cùng, các blockchain khác như Algorand và Internet Computer cung cấp một khoản phí cố định phụ thuộc vào giá token của họ (tương ứng là 0,001 ALGO và 0,0001 ICP).

2.4.5. Smart Contract



Hệ sinh thái của các nền tảng blockchain phát triển với một tốc độ khác nhau. Đối với một số có thể mất vài tháng để cho ra mắt bản cập nhật tiếp theo, trong khi những nền tảng khác như Internet Computer, đã có những bước phát triển lớn gần đây.

Kể từ khi Ethereum ra mắt Hợp đồng thông minh đầu tiên vào năm 2015, các blockchain khác đã làm theo. Một ví dụ rõ ràng là Cardano gần đây, thông qua "Alonzo Hard Fork", để tạo hợp đồng thông minh đầu tiên của mình mà không có thêm bất cứ cải tiến nào.

Hợp đồng thông minh của IC được gọi là 'Canisters' vì chúng là một gói mã WASM và Trang bộ nhớ, đồng thời chúng là sự phát triển và cải tiến của hợp đồng thông minh. Sự gia tăng đáng kể về số lượng của họ nói lên hoạt động ngày càng tăng của các nhà phát triển trên mạng:

Canisters loại bỏ các nút thắt cổ chai bằng cách sử dụng "Orthogonal Persistence", giúp loại bỏ nhu cầu duy trì và quản lý cơ sở dữ liệu bên ngoài hoặc khối lượng lưu trữ (code và dữ liệu lưu trữ trực tiếp on-chain). Trong khi các blockchain khác phải lưu trữ dữ liệu của họ trên các mạng lưu trữ phi tập trung khác.

Cộng đồng Internet Computer cũng đã thông qua đề xuất tăng dung lượng của Canisters từ 4GB lên 300GB. Rất ít ứng dụng cần nhiều dung lượng hơn, nhưng bạn có thể xây dựng dịch vụ/hệ thống của mình từ nhiều hợp đồng tùy thích nếu rơi vào trường hợp này.

Ngoài ra, một ngôn ngữ lập trình được gọi là Candid cho phép các canister tương tác với nhau bất kể ngôn ngữ lập trình mà chúng được phát triển.

Trong khi đó Cardano vẫn đang xem xét hợp đồng thông minh đầu tiên của mình.

IC sẽ tích hợp hợp đồng thông minh vào Bitcoin vào cuối năm nay. Có thể thông qua một ứng dụng của mật mã Khóa chuỗi sẽ tích hợp trực tiếp các mạng. Hợp đồng thông minh trên IC sẽ có thể giữ, gửi và nhận bitcoin mà không cần private key.

Các hợp đồng thông minh của Bitcoin sẽ chạy như các canister thông thường trên IC, do đó bạn sẽ tiết kiệm được rất nhiều chi phí giao dịch.

Đối với Ethereum, các nhà phát triển/dev trả tiền để triển khai các hợp đồng thông minh và mọi người trả tiền để sử dụng chúng nhưng đối với IC

thì mọi thứ hoàn toàn ngược lại. Các nhà phát triển sẽ là người trả phí cần thiết để chạy các ứng dụng/hợp đồng thông minh của họ (gọi là “cycles”). Hơn nữa trên Ethereum, để có thể lưu trữ 1GB dữ liệu, bạn có thể phải chi khoảng 5.000.000 USD nhưng đối với IC, 1GB có giá khoảng 3-5 USD.

Nói tóm lại, Canisters là Hợp đồng thông minh không có giới hạn cho phép tái tạo mọi thứ như web tương tác và dApps trên chuỗi (Blockchain Singularity) thay vì sử dụng dịch vụ của các công ty công nghệ lớn như AWS, Google, Azure, v.v.

2.4.6. Quản lý nhận dạng kĩ thuật số

Internet Computer xây dựng một hệ thống hoàn toàn mới mới cho việc quản lý danh tính người dùng thông qua hệ thống nhận dạng Internet – Internet Identity (II) mới của họ. Xác thực blockchain nâng cao này đảm bảo rằng dữ liệu của bạn không bị theo dõi hoặc bị khai thác. Nó cho phép bạn xác thực một cách an toàn và ẩn danh khi bạn truy cập các ứng dụng phi tập trung (dApps) sử dụng hệ thống xác thực.

Xác thực với các dịch vụ bằng cảm biến vân tay, Face ID, YubiKey, v.v.

Internet Identity đang được cải tiến liên tục để làm cho nó tương thích với ngày càng nhiều thiết bị. Hiện tại, nó cũng hỗ trợ Windows Hello như một phương thức xác thực.

Trong khi với các blockchain khác như Ethereum, người dùng cần ví bên ngoài như Metamask để tương tác với các ứng dụng phi tập trung. Dưới đây, bạn có thể thấy sự khác biệt giữa Ethereum và Internet Computer.

Dapps trên IC:

– Tạo danh tính – Internet Identity.

– Truy cập một trang web và sử dụng Dapp miễn phí.

– Hoặc bạn có thể sử dụng các ví được phát triển trên IC để ủy quyền – Ví Stoic và Ví Plug.

Dapps trên Ethereum :

– Sử dụng ví Metamask.

– Chuyển đến sàn giao dịch, tạo tài khoản, mua Ethereum.

– Gửi ETH đến Metamask.

– Truy cập một trang web, đăng nhập vào Metamask, sử dụng Dapp bằng cách thanh toán bằng ETH.

2.4.7. Quản trị mạng lưới /Bỏ phiếu /Đề xuất

IC sử dụng một hệ thống quản trị theo thuật toán được gọi là Hệ thống thần kinh mạng (NNS) cho phép chủ sở hữu ICP khóa các token để tạo ra "neurons". Những neuron này cung cấp quyền biểu quyết đối với các đề xuất được đưa ra, điều này ảnh hưởng đến hoạt động của mạng và mang lại cho người tham gia phần thưởng dưới dạng token ICP.

Trong số tất cả các blockchain được nêu trên, chỉ Polkadot và Avalanche có hệ thống quản trị, mặc dù trong trường hợp của Avalanche, chỉ dành cho các thông số mạng quan trọng (có nghĩa là họ đang đánh giá thấp quyền biểu quyết của cộng đồng). Chỉ một số thông số xác định trước có thể được sửa đổi thông qua quản trị, chẳng hạn như số tiền stake tối thiểu, tốc đúc token và các thông số kinh tế khác.

Ngoài ra, hệ thống quản trị của Algorand vẫn đang được phát triển.

2.4.8. Phần thưởng staking

Stake là quá trình ủy quyền hoặc khóa các token người dùng nắm giữ để kiếm phần thưởng. Khi bạn đã stake tài sản của mình, bạn có thể kiếm được phần thưởng dựa trên số tài đã stake và tăng lượng token đã stake bằng cách cộng gộp những phần thưởng đó trong tương lai.

Như bạn có thể thấy trong bảng trên, Internet Computer mang lại lợi nhuận cao nhất, dao động từ 15,4% mỗi năm đối với thời gian stake 6 tháng đến 28,9% mỗi năm đối với đặt cược 8 năm.

Tham khảo Máy tính ICP Neuron để xác định lợi nhuận bạn sẽ nhận được theo mục tiêu của mình. Các hướng dẫn sau đây cung cấp cho bạn hướng dẫn step-by-step giải thích làm thế nào để stake ICP sử dụng mạng thần kinh hệ thống (NNS).

2.5. Tổng kết

Internet Computer là một cách mới để lưu trữ các ứng dụng trên blockchain cho phép quyền sở hữu, an toàn hơn, khả năng chống kiểm duyệt cao hơn và dễ sử dụng hơn cho các nhà phát triển. Có rất nhiều cải tiến kỹ thuật blockchain (mật mã khóa chuỗi) mang lại những lợi ích này.

Internet Computer sử dụng mã thông báo tiện ích được gọi là ICP, có thể được đặt bên trong một nơ-ron để bỏ phiếu cho các đề xuất NNS và nhận phần thưởng đặt cược và được chuyển đổi thành các chu kỳ để sử dụng cho chi phí lưu trữ, tính toán và lưu trữ.

Internet Computer hiện vẫn là ‘người mới’, cung cấp cho thế giới một mô hình và công nghệ mới với các mục tiêu mang tính cách mạng trong thời gian tới như tích hợp Bitcoin và Ethereum .

Đây là blockchain nhanh nhất với thời gian xác thực là 2s và lệnh gọi truy vấn là 0.1s. Các hợp đồng thông minh trong canister của nó cung cấp một web 3.0 thực sự phục vụ web và tương tác trực tiếp với người dùng. Khả năng mở rộng là không giới hạn và IC cung cấp một blockchain có khả năng thích ứng cao cho phép cộng đồng bỏ phiếu cho các đề xuất thông qua Hệ thống thần kinh mạng để quản lý IC.

CHƯƠNG 3. CÀI ĐẶT CHƯƠNG TRÌNH

3.1. Opend

3.1.1. Những thuộc tính

```
private type Listing = {
    itemOver: Principal;
    itemPrice: Nat;
};

var mapOfNFTs = HashMap.HashMap<Principal, NFTActorClass.NFT>(1,Principal.equal,  Principal.hash);
var mapOfOwners = HashMap.HashMap<Principal, List.List<Principal>>(1, Principal.equal, Principal.hash);
var mapOfListings = HashMap.HashMap<Principal, Listing>(1, Principal.equal, Principal.hash);
```

Đoạn mã này khởi tạo 3 biến khác nhau:

mapOfNFTs là một bảng băm (hash map) được khởi tạo với 1 phần tử trống, và sử dụng Principal.equal và Principal.hash làm hàm so sánh và hàm băm cho các khóa (key) trong bảng băm.

mapOfOwners cũng là một bảng băm khởi tạo với 1 phần tử trống, và sử dụng cùng hai hàm Principal.equal và Principal.hash như trên.

mapOfListings là một bảng băm khởi tạo với 1 phần tử trống, và cũng sử dụng cùng hai hàm Principal.equal và Principal.hash như trên.

Đối tượng Listing là một kiểu dữ liệu định nghĩa trong đoạn mã, và chứa 2 thuộc tính: itemOver là một đối tượng Principal, và itemPrice là một số tự nhiên (Natural Number).

Các bảng băm này có thể được sử dụng để lưu trữ các thông tin về các đối tượng NFT (non-fungible token) và các người sở hữu chúng, và các danh sách cung cấp các NFT.

3.1.2. Phương thức mint

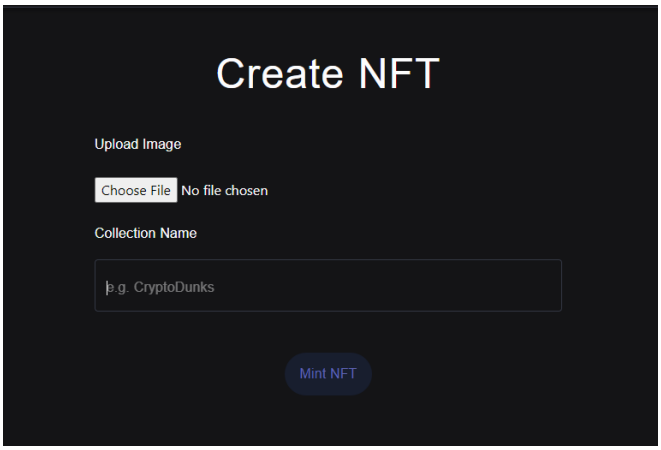
a) Back end

```
public shared(msg) func mint(imgData: [Nat8], name: Text) : async Principal {
    let owner : Principal = msg.caller;

    let newNFT = await NFTActorClass.NFT(name, owner, imgData);

    let newNFTPrincipal = await newNFT.getCanisterId();

    mapOfNFTs.put(newNFTPrincipal, newNFT);
    addToOwnershipMap(owner, newNFTPrincipal);
    return newNFTPrincipal;
};
```

Hàm mint và có 2 tham số đầu vào: imgData là một mảng các số tự nhiên 8 bit, và name là một chuỗi ký tự (Text). Hàm này trả về một Principal bất đồng bộ (async).

Trong hàm này, biến owner được gán bằng người gọi hàm, và một đối tượng NFT mới được tạo ra bằng cách gọi hàm NFTActorClass.NFT với các tham số name, owner, và imgData. Sau đó, một Principal mới được tạo ra bằng cách gọi hàm getCanisterId trên đối tượng NFT mới tạo. Cuối cùng, Principal mới được thêm vào bảng băm mapOfNFTs với NFT mới tạo làm giá trị, và hàm addToOwnershipMap được gọi với tham số owner và newNFTPrincipal để cập nhật bảng băm mapOfOwners. Cuối cùng, hàm trả về Principal mới

b) Front end



Giao diện tạo mới 1 NFT

```
<span onClick={handleSubmit(onSubmit)} className="form-Chip-label">Mint NFT</span>
```

```
async function onSubmit(data) {
  setLoaderHidden(false);
  const name = data.name;
  const image = data.image[0];
  const imageArray = await image.arrayBuffer();
  const imageByteData = [...new Uint8Array(imageArray)];
  const newNFTID = await opend.mint(imageByteData, name);

  setNFTPrincipal(newNFTID);
  setLoaderHidden(true)
}
```

Trong hàm này, hàm setLoaderHidden được gọi với đối số false. Sau đó, các biến name và image được khởi tạo bằng cách truy cập các thuộc tính

tương ứng trong đối tượng data. Biến imageArray được khởi tạo bằng cách gọi hàm arrayBuffer trên đối tượng image, và biến imageByteData được khởi tạo bằng một mảng các phần tử 8 bit từ imageArray. Sau đó, hàm mint được gọi trên đối tượng opend với các tham số imageByteData và name. Kết quả trả về của hàm mint được gán cho biến newNFTID. Tiếp theo, hàm setNFTPrincipal được gọi với tham số newNFTID, và hàm setLoaderHidden được gọi với đối số true để ẩn loader.

3.1.3. Phương thức addtoOwnershipmap

```
private func addToOwnershipMap(owner: Principal, nftId: Principal) {
    var ownedNFTs : List.List<Principal> = switch (mapOfOwners.get(owner)) {
        case null List.nil<Principal>();
        case (?result) result;
    };

    ownedNFTs := List.push(nftId, ownedNFTs);
    mapOfOwners.put(owner, ownedNFTs);
};
```

Hàm addToOwnershipMap và có 2 tham số đầu vào: owner là một đối tượng Principal, và nftId là một đối tượng Principal. Hàm này không có giá trị trả về (void).

Trong hàm này, biến ownedNFTs được khởi tạo bằng cách sử dụng câu lệnh switch-case (switch statement) để truy cập giá trị trong bảng băm mapOfOwners với khóa là owner. Nếu không tìm thấy giá trị nào, biến ownedNFTs được khởi tạo bằng một danh sách rỗng (empty list). Nếu tìm thấy giá trị, biến ownedNFTs được gán bằng giá trị đó.

Sau đó, biến ownedNFTs được cập nhật bằng cách sử dụng hàm List.push để thêm nftId vào cuối danh sách, và bảng băm mapOfOwners được cập nhật bằng cách gọi hàm put với khóa là owner và giá trị là ownedNFTs.

3.1.4. Phương thức getOwnedNFTs, getListedNFTs

a) Back end

```
public query func getOwnedNFTs(user: Principal) : async [Principal] {
    var userNFTs: List.List<Principal> = switch (mapOfOwners.get(user)) {
        case null List.nil<Principal>();
        case (?result) result;
    };

    return List.toArray(userNFTs);
};
```

Hàm này có 1 tham số đầu vào là user là một đối tượng Principal, và trả về một mảng các đối tượng Principal bất đồng bộ (async).

Trong hàm này, biến userNFTs được khởi tạo bằng cách sử dụng câu lệnh switch-case (switch statement) để truy cập giá trị trong bảng băm mapOfOwners với khóa là user. Nếu không tìm thấy giá trị nào, biến userNFTs được khởi tạo bằng một danh sách rỗng (empty list). Nếu tìm thấy giá trị, biến userNFTs được gán bằng giá trị đó.

Cuối cùng, hàm trả về một mảng các đối tượng Principal được tạo từ danh sách userNFTs bằng cách gọi hàm List.toArray.

```
public query func getListedNFTs() : async [Principal] {
    let ids = Iter.toArray(mapOfListings.keys());
    return ids;

};
```

Trong hàm này, biến ids được khởi tạo bằng một mảng các khóa trong bảng băm mapOfListings được tạo bởi hàm Iter.toArray. Cuối cùng, mảng ids được trả về như là kết quả của hàm.

b) Front end

```
async function getNFTs() {
  const userNFTIds = await opend.getOwnedNFTs(CURRENT_USER_ID);
  console.log(userNFTIds);
  setUserOwnedGallery(<Gallery title="My NFTs" ids={userNFTIds} role="collection"/>)

  const listedNFTs = await opend.getListedNFTs();
  console.log(listedNFTs);
  setListingGallery(<Gallery title={"Discover"} ids={listedNFTs} role="discover"/>)
}
```

Trong hàm này, biến userNFTIds được khởi tạo bằng kết quả trả về của hàm getOwnedNFTs gọi trên đối tượng opend với đối số là CURRENT_USER_ID. Hàm setUserOwnedGallery được gọi với component Gallery được truyền các props title, ids, và role. Cuối cùng, biến listedNFTs được khởi tạo bằng kết quả trả về của hàm getListedNFTs gọi trên đối tượng opend và hàm setListingGallery được gọi tương tự như trên với biến listedNFTs làm đối số.

3.1.5. Phương thức listItem

a) Back end

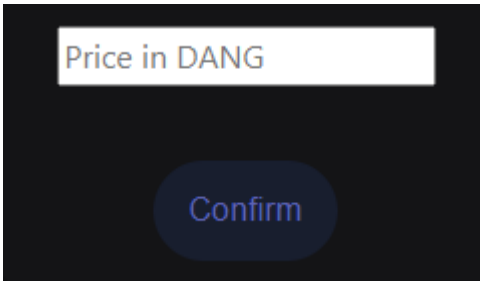
```
  public shared(msg) func listItem(id: Principal, price: Nat) : async Text {
      var item : NFTActorClass.NFT = switch (mapOfNFTs.get(id)) {
          case null return "NFT does not exist.";
          case (?result) result;
      };

      let owner = await item.getOwner();
      if(Principal.equal(owner, msg.caller) ) {
          let newListing : Listing = {
              itemOver = owner;
              itemPrice = price;
          };
          mapOfListings.put(id, newListing);
          return "Sucess";
      } else {
          return "You don't have the NFT."
      }

  };
```

Hàm này có một biến "item" được khởi tạo bằng cách sử dụng câu lệnh "switch" để kiểm tra xem biến "id" có tồn tại trong "mapOfNFTs" hay không.

Nếu không, hàm sẽ trả về chuỗi "NFT does not exist."; Nếu có, hàm sẽ tiếp tục thực hiện các lệnh tiếp theo. Sau đó, hàm sẽ khởi tạo biến "owner" bằng cách gọi hàm "getOwner" trên biến "item".

Nếu "owner" trùng với người gọi hàm ("msg.caller"), hàm sẽ khởi tạo một đối tượng "newListing" với hai thuộc tính "itemOver" và "itemPrice" được gán bằng "owner" và "price" tương ứng, sau đó lưu trữ đối tượng này trong biến "mapOfListings" với “id” là owner của NFT và trả về chuỗi "Success". Nếu "owner" không trùng với người gọi hàm, hàm sẽ trả về chuỗi "You don't have the NFT.".

b) Front end



Giao diện input và button để bán 1 NFT

```
async function sellItem() {
  setBlur({filter: "blur(4px)" })
  setLoaderHandle(false);
  console.log("set price = " + price);
  const listingResult = await opend.listItem(props.id, Number(price));
  console.log("listing: " + listingResult);
  if(listingResult == "Sucess") {
    const openDId = await opend.getOpenDCanisterID();
    const transferResult = await NFTActor.transferOwnership(openDId);
    console.log("transfer : " + transferResult);
    if( transferResult == "Success") {
      setLoaderHandle(true);
      setButton();
      setPriceInput();
      setOwner("OpenD");
      setSellStatus("Listed")

    }
  }
}
```

Hàm này sẽ thực hiện một số công việc như sau:

1. Sử dụng câu lệnh "setBlur" để thiết lập giá trị cho một thuộc tính có tên "filter" bằng "blur(4px)".

2. Sử dụng câu lệnh "setLoaderHandle" để thiết lập giá trị cho một biến bằng "false".

3. Gọi hàm "listItem" trên đối tượng "opend" với hai tham số "props.id" và "Number(price)". Kết quả trả về từ hàm này được lưu trữ trong biến "listingResult".

4. Nếu giá trị của "listingResult" là "Success", hàm sẽ tiếp tục thực hiện các lệnh tiếp theo:

a. Khởi tạo biến "openDId" bằng kết quả trả về từ hàm "getOpenDCanisterID" trên đối tượng "opend".

b. Gọi hàm "transferOwnership" trên đối tượng "NFTActor" với tham số "openDId". Kết quả trả về từ hàm này được lưu trữ trong biến "transferResult".

5. Nếu giá trị của "transferResult" là "Success", hàm sẽ sử dụng các câu lệnh "setLoaderHandle", "setButton", "setPriceInput", "setOwner" và "setSellStatus" để thiết lập các giá trị cho các biến tương ứng.

3.1.6. Phương thức completePurchase

a) Back end

```
public shared(msg) func completePurchase(id: Principal, ownerId: Principal, newOwnerId: Principal) : async Text {
    var purchasedNFT : NFTActorClass.NFT = switch (mapOfNFTs.get(id)) {
        case null return "NFT does not exist";
        case (?result) result;
    };

    let transferResult = await purchasedNFT.transferOwnership(newOwnerId);
    if(transferResult == "Success") {
        mapOfListings.delete(id);
        var ownedNFTs : List.List<Principal> = switch (mapOfOwners.get(ownerId)) {
            case null List.nil<Principal>();
            case (?result) result;
        };
        ownedNFTs := List.filter(ownedNFTs, func(listItemId: Principal) : Bool {
        return listItemId != id;
        });
        addToOwnershipMap(newOwnerId, id);
        return "Success";
    }
    else {
        return "Error";
    }
}
```

Phương thức này được định nghĩa với tên "completePurchase" và nhận ba tham số: id (kiểu Principal), ownerId (kiểu Principal) và newOwnerId (kiểu Principal). Nó trả về một kết quả kiểu async Text.

Trong phương thức này, có một biến có tên "purchasedNFT" được khởi tạo bằng kết quả trả về bởi một lời gọi đến phương thức "get" của một đối tượng "mapOfNFTs" và truyền vào tham số "id". Nếu kết quả trả về là null, biến "purchasedNFT" sẽ được gán bằng chuỗi "NFT does not exist". Nếu không, nó sẽ được gán bằng kết quả trả về.

Sau đó, phương thức này gọi phương thức "transferOwnership" của đối tượng "purchasedNFT" và truyền vào tham số "newOwnerId". Kết quả trả về được lưu trữ trong biến "transferResult". Nếu "transferResult" bằng "Success", phương thức "delete" của đối tượng "map OfListings" được gọi và truyền vào tham số "id" để xóa một mục từ danh sách.

Sau đó, biến "ownedNFTs" được khởi tạo bằng kết quả trả về bởi một lời gọi đến phương thức "get" của đối tượng "mapOfOwners" và truyền vào tham số "ownerId". Nếu kết quả trả về là null, biến "ownedNFTs" sẽ được gán bằng một danh sách trống. Nếu không, nó sẽ được gán bằng kết quả trả về.

Sau đó, biến "ownedNFTs" được cập nhật bằng kết quả trả về từ lời gọi đến phương thức "filter" của đối tượng "List" và truyền vào biến "ownedNFTs" và một hàm với một tham số "listItemId" (kiểu Principal). Hàm này sẽ trả về giá trị true nếu "listItemId" không bằng "id" và ngược lại.

Cuối cùng, phương thức "addToOwnershipMap" được gọi và truyền vào tham số "newOwnerId" và "id". Nếu không có lỗi xảy ra trong quá trình thực hiện, phương thức sẽ trả về chuỗi "Success". Nếu có lỗi xảy ra, nó sẽ trả về chuỗi "Error".

```
async function handleBuy() {
  console.log("Buy was triggered");
  setLoaderHandle(false);
  const tokenActor = await Actor.createActor(tokenIdFactory, {
    agent,
    canisterId: Principal.fromText("xpeh5-6iaaa-aaaaa-aaaua-cai")
  })

  const sellerId = await opend.getOriginalOwner(props.id);
  const itemPrice = await opend.getListedNFTPrice(props.id);

  const result = await tokenActor.transfer(sellerId, itemPrice);
  if (result == "Success") {
    const transferResult = await opend.completePurchase(props.id, sellerId, CURRENT_USER_ID)
    console.log("Purchase: " + transferResult)
    setLoaderHandle(true);
    setDisplay(false)
  }
}
```

Hàm này được định nghĩa với tên "handleBuy" và không nhận tham số. Nó được đánh dấu với từ khóa "async" nên có thể chứa các lời gọi hàm bất đồng bộ.

Trong hàm này, hàm "setLoaderHandle" được gọi và truyền vào đối số "false" để thiết lập giá trị của một biến. Sau đó, biến "tokenActor" được khởi tạo bằng kết quả trả về bởi lời gọi đến hàm "createActor" của đối tượng "actor" và truyền vào hai tham số: "tokenIdFactory" và một đối tượng chứa các thuộc tính "agent" và "canisterId".

Sau đó, biến "sellerId" được khởi tạo bằng kết quả trả về bởi lời gọi đến hàm "getOriginalOwner" của đối tượng "opend" và truyền vào tham số "props.id". Biến "itemPrice" được khởi tạo bằng kết quả trả về bởi lời gọi đến hàm "getListedNFTPrice" của đối tượng "opend" và truyền vào tham số "props.id".

Sau đó, hàm "transfer" của đối tượng "tokenActor" được gọi và truyền vào hai tham số: "sellerId" và "itemPrice". Kết quả trả về được lưu trữ trong biến "result".

Nếu "result" bằng "Success", hàm "completePurchase" của đối tượng "opend" được gọi và truyền vào ba tham số: "props.id", "sellerId" và "CURRENT_USER_ID". Kết quả trả về được in ra màn hình bằng lời gọi đến hàm "console.log" và sau đó hàm "setLoaderHandle" được gọi với đối số "true" và hàm "setDisplay" được gọi với đối số "false" để thiết lập giá trị của hai biến “loaderHandle” và “display”.

3.1.7 Một số phương thức get khác

```
public query func getOpenDCanisterID() : async Principal {
    return Principal.fromActor(OpenD);
};
```

```
public query func isListed(id: Principal) : async Bool {
    if(mapOfListings.get(id) == null) {
        return false;
    } else {
        return true;
    }
};
```

```
public query func getOriginalOwner(id: Principal) : async Principal {
    var listing : Listing = switch (mapOfListings.get(id)) {
        case null return Principal.fromText("");
        case (?result) result;
    };

    return listing.itemOver
};
```

```
public query func  getListedNFTPrice(id: Principal) : async Nat {
    var listing : Listing = switch (mapOfListings.get(id)) {
        case null return 0;
        case (?result) result;
    };
    return listing.itemPrice;
};
```

3.2. Actor class NFT

3.2.1. Các thuộc tính

```
private let itemName = name;
private var nftOwner = owner;
private let imageBytes = content;
```

Biến "itemName" là một biến private có kiểu dữ liệu Text và được khởi tạo bằng tham số "name".

Biến "nftOwner" là một biến private có kiểu dữ liệu Principal và được khởi tạo bằng tham số "owner".

Biến "imageBytes" là một biến private có kiểu dữ liệu mảng các số nguyên tố 8 bit và được khởi tạo bằng tham số "content".

3.2.2. Các phương thức get

```
public query func getName() : async Text {
    return itemName;
};
public query func getOwner() : async Principal {
    return nftOwner;
};
public query func getAsset() : async [Nat8] {
    return imageBytes;
};
public query func getCanisterId() : async Principal {
    return Principal.fromActor(this);
};
```

Phương thức "getName" trả về giá trị của biến "itemName" có kiểu dữ liệu Text. Phương thức "getOwner" trả về giá trị của biến "nftOwner" có kiểu dữ liệu Principal. Phương thức "getAsset" trả về giá trị của biến "imageBytes" có kiểu dữ liệu mảng các số nguyên tố 8 bit.

Phương thức "getCanisterId" trả về Canister ID của lớp hiện tại. Canister ID là một định danh duy nhất được sử dụng để xác định một Canister (tức là một tập hợp các lớp và dữ liệu) trong hệ thống. Canister ID là một định danh Principal, tức là một định danh của một đối tượng trong hệ thống. Phương thức "fromActor" được sử dụng để tạo ra một định danh Principal từ một đối tượng (tức là lớp "NFT" hiện tại).

3.2.3. Phương thức transferOwnership

```
public shared(msg) func transferOwnership(newOwner: Principal) : async Text {
    if(msg.caller == nftOwner) {
        nftOwner := newOwner;
        return "Success";
    } else {
        return "Error: Not initated by NFT Owner"
    }
}
```

Trong phương thức này, có một điều kiện được kiểm tra để xác định xem gọi hàm này có được thực hiện bởi chủ sở hữu hiện tại của NFT hay không. Nếu đúng, giá trị của biến "nftOwner" sẽ được cập nhật thành giá trị của tham số "newOwner" và phương thức sẽ trả về chuỗi "Success". Nếu không, phương thức sẽ trả về chuỗi "Error: Not initiated by NFT Owner".

Biến "msg" là một biến có sẵn trong mọi phương thức shared và chứa thông tin về gọi hàm này. Trong trường hợp này, biến "msg.caller" chứa định danh của người gọi hàm này. Nếu định danh này khớp với giá trị của biến "nftOwner", điều kiện sẽ được thỏa mãn và phương thức sẽ thực hiện các lệnh trong khối if. Nếu không, phương thức sẽ thực hiện các lệnh trong khối else.

3.3. Actor Token

3.3.1. Các thuộc tính

```
let owner : Principal = Principal.fromText("26cgp-iaht5-dxxyb-beyke-lvrup-nzdjq-zizqj-pydki-udzde-obomn-rae");
let totalSupply : Nat = 1000000000;
let symbol : Text = "DANG";
```

Biến "owner" là tài khoản chủ sở hữu của actor.

Biến "totalSupply" là tổng số lượng đồng tiền có sẵn trong actor.

Biến "symbol" là ký hiệu của đồng tiền.

Biến "owner" được khởi tạo với giá trị là một tài khoản được tạo bằng cách sử dụng hàm "Principal.fromText". Biến "totalSupply" và "symbol" được khởi tạo với giá trị cố định.

3.3.2. Phương thức payOut

```
public shared(msg) func payOut() : async Text {

  Debug.print(debug_show(msg.caller));
  if (balances.get(msg.caller) == null) {
    let amount = 10000;
    let result = await transfer(msg.caller, amount);
    return result;
  } else {
    return "Already Claimed";
  }
};
```

Trong hàm này, hệ thống sẽ kiểm tra xem tài khoản gọi hàm có tồn tại trong bảng băm "balances" hay không. Nếu không tồn tại, hệ thống sẽ gọi hàm "transfer" để chuyển 10000 đồng tiền cho tài khoản gọi hàm, và trả về kết quả thực hiện hàm "transfer". Nếu tài khoản gọi hàm đã tồn tại trong bảng băm "balances", hàm sẽ trả về chuỗi "Already Claimed".

```
async function handleClick(event) {
  setDisable(true);
  const result = await token.payOut();
  console.log("payout: " + result);
  setText(result);
}
```

Hàm này sử dụng từ khóa async để chỉ rằng nó sẽ chạy một cách bất đồng bộ, có nghĩa là nó sẽ không chặn luồng chương trình khi chờ đợi kết quả của các hàm bất đồng bộ khác.

Hàm handleClick có một tham số đầu vào là event, dùng để lấy thông tin về sự kiện bấm nút được gọi. Trong hàm này, nó gọi hàm setDisable(true) để đặt trạng thái của nút bấm là không hoạt động, và gọi hàm token.payOut() để thực hiện giao dịch. Kết quả của hàm payOut được lưu trữ trong biến result. Cuối cùng, nó gọi hàm setText(result) để đặt nội dung của kết quả làm nội dung của một phần tử HTML.

3.3.3. Phương thức transfer

```
public shared(msg) func transfer(to: Principal, amount: Nat) : async Text {
  let fromBalance = await balanceOf(msg.caller);
  if (fromBalance > amount) {
    let newFromBalance : Nat = fromBalance - amount;
    balances.put(msg.caller, newFromBalance);

    let toBalance = await balanceOf(to);
    let newToBalance = toBalance + amount;
    balances.put(to, newToBalance);

    return "Success";
  } else {
    return "Insufficient Funds"
  }
};
```

Trong hàm này, hệ thống sẽ lấy số dư hiện tại của tài khoản gọi hàm bằng cách sử dụng hàm "balanceOf". Nếu số dư có số lượng lớn hơn số lượng đồng tiền cần chuyển, hệ thống sẽ trừ số lượng đồng tiền tương ứng khỏi số dư của tài khoản gọi hàm, và cộng số lượng đồng tiền tương ứng vào số dư của tài khoản nhận. Sau khi thực hiện chuyển tiền thành công, hàm sẽ trả về chuỗi "Success". Nếu số dư không đủ để thực hiện chuyển tiền, hàm sẽ trả về chuỗi "Insufficient Funds".

3.4. React router dom

```
<Switch>
  <Route exact path="/">
    <img className="bottom-space" src={homeImage} />
  </Route>
  <Route path="/discover">
    {listingGallery}
  </Route>
  <Route path="/minter">
    <Minter/>
  </Route>
  <Route path="/collection">
    {userOwnedGallery}
  </Route>
</Switch>
```

Mỗi thành phần Route định nghĩa một đường dẫn khác nhau và nội dung mà nó sẽ hiển thị khi đường dẫn đó được truy cập.

Ví dụ, khi đường dẫn là "/", thì thành phần Route sẽ hiển thị một hình ảnh homeImage. Khi đường dẫn là "/discover", thành phần Route sẽ hiển thị giá trị của biến listingGallery. Khi đường dẫn là "/minter", thành phần Route sẽ hiển thị một thành phần Minter. Và khi đường dẫn là "/collection", thành phần Route sẽ hiển thị giá trị của biến userOwnedGallery.

Thành phần Switch sẽ chỉ hiển thị một thành phần Route duy nhất trong số những thành phần Route được định nghĩa bên trong nó, và nó sẽ chọn thành phần Route đầu tiên mà đường dẫn tương ứng với đường dẫn hiện tại.

```
<button className="ButtonBase-root Button-root Button-text header-navButtons-3">
  <Link to="/discover">Discover</Link>
</button>
<button className="ButtonBase-root Button-root Button-text header-navButtons-3">
<Link to="/minter">Minter</Link>
</button>
<button className="ButtonBase-root Button-root Button-text header-navButtons-3">
<Link to="/collection">My NFTs</Link>
</button>
```

Các thẻ Link được gắn các đường dẫn

KẾT LUẬN

Qua quá trình nghiên cứu về blockchain và một số ứng dụng của công nghệ này, cùng với sự giúp đỡ tận tình của thầy cô và bạn bè, luận văn đã đạt được một số kết quả nhất định, đưa ra cái nhìn cụ thể hơn về khái niệm blockchain, Internet Computer và Smart Contract và ứng dụng của nó trong mảng minting và giao dịch NFT.

Về mặt nội dung, luận văn đã đạt được một số kết quả sau đây:

1. Tìm hiểu và nghiên cứu lý thuyết:

• Chi tiết về công nghệ blockchain

• Internet Computer và Smart Contract

• Mô hình ứng dụng blockchain

2. Thực nghiệm:

• Hoàn thành chương trình tạo một blockchain, đúc và trao đổi NFT

Phân công nhiệm vụ

Ngô Văn Thuận	Xây dựng Front end của Website
Nguyễn Minh Phú	Xây dựng Back end NFT
Nguyễn Ngọc Thanh	Xây dựng Back end Token
Nguyễn Văn Mạnh Duy	Xây dựng Back end Opened

Link github: https://github.com/Manhduy1711/block-chain-OpenD