UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

# Estudi del sistema de fitxers distribuït

# InterPlanetary File System

**Treball de Fi de Grau**

**presentat a la Facultat Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona**

**Universitat Politècnica de Catalunya**

**by**

**Jordi Salarich i Martínez**

**En compliment amb els requisits**

**per a la titulació del**

*GRAU EN ENGINYERIA DE TECNOLOGIES I SERVEIS DE TELECOMUNICACIÓ*

**Director: Jose Luis Muñoz Tapia**

**Barcelona, Gener 2019**

# Índex


**1 IPFS** **5**

1.1 Introducció . . . 5
1.1.1 Resum . . . 5
1.1.2 Resumen . . . 5
1.2 Introducció als sistemes Peer to Peer . . . 5
1.2.1 Sistemes centralitzats . . . 6
1.2.2 Sistemes Descentralitzats . . . 6
1.2.3 Sistemes Híbrids . . . 7
1.2.4 Sistemes Distribuïts . . . 7
1.3 Introducció a IPFS - InterPlanetary File System . . . 8
1.3.1 Protocol HTTP . . . 8
1.4 Tecnologies relacionades . . . 11
1.4.1 Taula de Hash Distribuïda - DHT . . . 11
1.4.2 Kademlia DHT . . . 12
1.4.3 S/Kademlia . . . 14
1.4.4 Coral DSHT . . . 14
1.4.5 Merkle Direct Acyclic Graph (DAG) . . . 16
1.4.6 Bitswap . . . 19
1.5 IPFS en profunditat . . . 20
1.5.1 Fingerprinting . . . 20
1.5.2 MultiHash . . . 20
1.5.3 Verificació . . . 20
1.5.4 Routing - DHT . . . 21
1.5.5 Pinning . . . 21
1.5.6 IPNS - Inter Planterary Name System . . . 23
1.5.7 Path . . . 23
1.5.8 PubSub . . . 24
1.5.9 Seguretat - Encriptar fitxers pujats a la web . . . 24
1.6 Tutorials - Learning by doing . . . 25
1.6.1 IPFS desde zero . . . 25
1.6.2 Path . . . 27
1.6.3 Online - Daemon . . . 28

1.6.4 Get i Cat . . . . . 28
1.6.5 DHT . . . . . 29
1.6.6 Pinning i ús del Garbage Collector . . . . . 30
1.6.7 Bitswap . . . . . 30
1.6.8 PubSub En línea . . . . . 31
1.7 Sistemes relacionats . . . . . 32
1.7.1 Filecoin . . . . . 32
1.8 Conclusions i desenvolupament futur . . . . . 34
1.8.1 Idees pera desenvolupar . . . . . 34
1.9 Budget . . . . . 35
1.9.1 Cost Recurs Humà . . . . . 35

# Capítol 1

# IPFS

## 1.1 Introducció

### 1.1.1 Resum

En el següent Treball de Fi de Grau s'introdueix un estudi detallat del sistema distribuït de fitxers InterPlanetary File System, dissenyat i implementat per a l'empresa Protocol Labs, que permet emmagatzemar arxius a la xarxa sense la necessitat d'un servidor central amb el suport de taules de hash distribuïdes (Kademlia DHT) que s'expliquen en detall. Alhora, s'introdueix un tutorial per a conèixer, comprendre els conceptes i poder utilitzar IPFS d'una forma més pràctica.

### 1.1.2 Resumen

En el siguiente Trabajo de Fin de Grado se introduce un estudio detallado del sistema distribuido de ficheros Interplanetary File System, diseñado e implementado por la empresa Protocol Labs, que permite el almacenamiento de archivos en la red sin necesidad de un servidor central con el apoyo de tablas de hash distribuidas (Kademlia DHT) que se explican en detalle. Asimismo, se introduce un tutorial para conocer, comprender los conceptos y poder utilizar IPFS de una forma más práctica.

## 1.2 Introducció als sistemes Peer to Peer

Els Sistemes Peer to Peer [2], coneguts també com a sistemes P2P o d' igual a igual, són sistemes de comunicació entre nodes que actuen com a client - servidor de forma conjunta, convertint així els dispositius en elements actius que permeten als usuaris intercanviar informació i/o agrupar capacitat de processament sense la necessitat de tenir un servidor central.

Per tal d'entrar en detall al Sistema de Fitxers Distribuït IPFS que es tractarà al llarg del document, s'explicaran els diversos tipus de sistemes existents. Començant per els més clàssics com són els sistemes centralitzats, descentralitzats i híbrids, fins a les noves estructures distribuïdes que existeixen anomenats sistemes distribuïts P2P.

### 1.2.1 Sistemes centralitzats

En un sistema centralitzat tots els nodes són perifèrics excepte el servidor central que permet la interconnexió i transferència d'informació entre els nodes i els seus canals. Per a aconseguir-ho, el node central mapeja els recursos identificant dels nodes que emmagatzemen aquests recursos.
Tot i identificar diversos punts de confort i facilitats, existeixen parts crítiques així com la vulnerabilitat de tenir un punt central a atacar per a obtenir la informació de la xarxa o un problema d'escalabilitat si molts usuaris necessiten obtenir informació - del servidor central - i creen un coll d'ampolla o "bottleneck"fent disminuir així la velocitat de la xarxa o fins i tot bloquejant-la de forma completa.

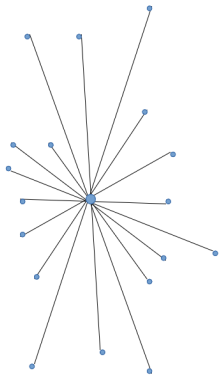

Figura 1.1: Sistema centralitzat.

Tot i que és un dels sistemes més clàssics i antics, continua sent el sistema més utilitzat fins al moment. Algun dels camps als quals està present és en la televisió, la qual emet la informació des de un punt central als receptors que tenen els usuaris, en els diaris o també en models de política dels estats centralistes.

### 1.2.2 Sistemes Descentralitzats

A l'extrem contrari dels sistemes centralitzats i trobem els sistemes descentralitzats, els quals no tenen un únic punt central per a obtenir la informació d'interès sinó que hi ha un centre col·lectiu de diversos ports de connexió. La caiguda d'un dels nodes "pare"que forma part del centre col·lectiu no provoca la desconnexió de tota la xarxa sinó solament de la xarxa regulada pel node caigut.
Els nodes es connecten entre ells sense la necessitat de tenir una connexió directa amb un òrgan central.
Els inconvenients que tenen aquest tipus de sistemes estan relacionats en:
- No hi ha garantia de trobar un determinat recurs existent a la xarxa pel fet que no existeix un òrgan central que indexi on es pot trobar tot el contingut.
- No és un sistema escalable.

Existeixen diversos exemples de sistemes descentralitzats com és la Wikipèdia o també softwares com Gnutella o Freenet que utilitzen xarxes descentralitzades per a la distribució d'informació.

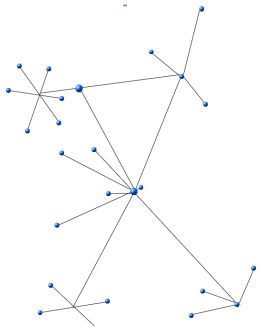

Figura 1.2: Sistema descentralitzat.

### 1.2.3 Sistemes Híbrids

Un conjunt de sistemes, o sistema híbrid, és la combinació de diversos sistemes. Per a aconseguir-ho hi ha diversos nodes amb funcionalitats privilegiades (com el node central en els sistemes centralitzats) amb capacitat de realitzar consultes a un servidor per a saber on trobar els altres nodes de la xarxa.
Alguns dels softwares que utiltitzen sistemes híbrids són eDonkeym, BitTorrent o Skype.

### 1.2.4 Sistemes Distribuïts

Els sistemes distribuïts o P2P s'estructuren en forma de xarxa sense la necessitat d'un òrgan central que intervingui en les comunicacions entre nodes. La clau fonamental dels sistemes P2P és que tots els nodes són tractats d'igual a igual sense privilegis a la xarxa i tota la informació no està allotjada a un sol punt, sinó que el conjunt de nodes tenen la informació d'interès emmagatzemada i són capaços d'obtenir la informació de la xarxa. Per a aconseguir-ho, s'utilitzen les Taules de Hash Distribuïdes, explicat en detall a l'apartat 1.4.1. Cada usuari és reponsable de una part específica del contingut de la xarxa, en el cas de IPFS cada usuari és responsable de les dades locals que emmagatzema i utilitza Kademlia-DHT, explicat en el següent apartat 1.4.2, tot i que en un futur proper té l'objectiu de millorar la seguretat adherint taules de Hash de S/Kademlia[4] i Coral DSHT [5].
Avantatges dels sistemes distribuïts:
- Tot node és igual als altres del conjunt del sistema, i al afegir un node nou no és necessari reestructurar tota la xarxa.
- Disminució de congestió i colls d'ampolla, ja que les connexions es realitzen punt a punt i no han d'anar a un òrgan central a buscar la informació.
- Escalabilitat.
- Autoorganització.
Inconvenients:
- Gestió més complexa, i per tant més cara amb majors possibilitats de tenir una xarxa insegura si no es tracta de forma correcta.
- És possible no obtenir un fitxer existent a la xarxa.

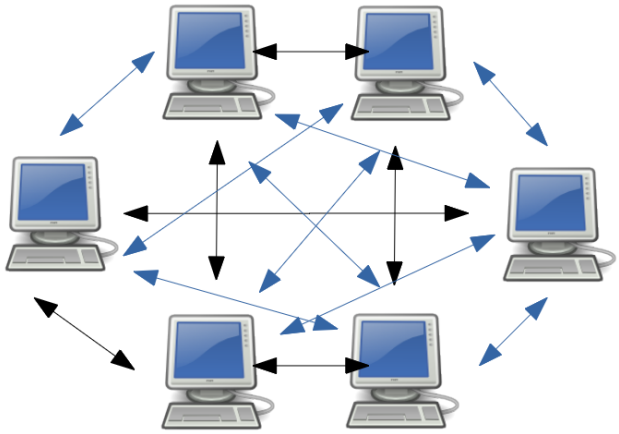

Figura 1.3: Sistema distribuït.

Alguns dels sistemes distribuïts més utilitzats a part de Kademlia-DHT son Chord, Pastry o Tulip Overlay.

## 1.3 Introducció a IPFS - InterPlanetary File System

InterPlanetary File System [1], conegut per les seves inicials IPFS, és un nou i ambiciós sistema d'arxius distribuït Peer to Peer (P2P) en constant desenvolupament i millora, creat a l'empresa Protocol Labs amb la col·laboració de la comunitat de IPFS d'arreu del món amb l'objectiu d'interconnectar tots els dispositius en un mateix sistema de fitxers, oferint una solució P2P, o xarxa entre iguals, en una estructura distribuïda sense la necessitat de tenir un servidor central per a obtenir un contingut.
Per a aconseguir-ho es basa en un conjunt d'idees i sistemes ja existents, com en el protocol per a l'intercanvi de blocs d'informació i distribució de fitxers semblant al que utilitza BitTorrent [7], conegut com BitSwap, el sistema de hosting de repositoris MerkleDAG que ofereix GitHub o també les Taules de Hash Distribuïdes (DHT - sigles en anglès de Distributed Hash Table) que s'han utilitzat per xarxes P2P com són Kademlia DHT entre altres conceptes detallats en els següents apartats.

### 1.3.1 Protocol HTTP

El protocol de distribució d'arxius a la xarxa per excel·lència és el protocol HTTP (Protocol de Transferència Hipertext). HTTP es basa en el concepte de client - servidor, on el client web es

comunica constantment amb el servidor per obtenir els fitxers d'interès i el servidor web l'hi respon oferint la informació sol·licitada com es mostra en la següent figura.

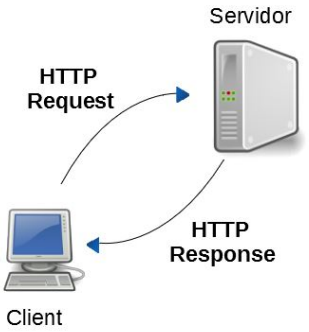


Figura 1.4: Model simple del funcionament del protocol HTTP

Un dels problemes que permet resoldre IPFS respecte HTTP amb l'ajut de l'adreçament a contingut és l'allotjament d'arxius en servidors llunyans i amb la necessitat de descarregar i obtenir informació del servidor constantment, aquest comportament provoca un consum molt elevat de l'ample de banda de la xarxa i augmenta la lentitud a la xarxa. Amb l'ajut de IPFS i amb les idees de BitTorrent es trenca aquesta barrera i s'afegeix la capacitat d'obtenir contingut provinent de diversos nodes, aconseguir-lo amb menys temps i de forma eficient. Un altre punt important en el qual IPFS es recolza i resolt és en l'estabilitat i permanència, gràcies l'adreçament de continguts amb "hashos criptogràfics" adreçats a contingut, és a dir únics per a cada arxiu pujat a la xarxa, es permet verificar les dades i discriminar nodes que no són de confiança. Una vegada obtenim un contingut o arxiu, el podem guardar de forma indefinida al nostre repositori local i utilitzar-lo/visualitzar-lo quan el necessitem.

Al treballar en xarxes distribuïdes, i trencar l'esquema dels sistemes centralitzats, IPFS és capaç de solucionar problemes de seguretat, fent gairebé impossibles els atacs DDoS sobre els servidors que emmagatzemen informació, ja que caldria atacar a tots els nodes que tenen un contingut concret per a fer caure el sistema de fitxers de IPFS.

Tot i observar diferents problemes en el protocol HTTP, fins ara s'han dedicat molts recursos a HTTP i s'ha intentat millorar constantment amb la modificació de HTTP a HTTP/2, i sembla complicat fer un canvi a un nou protocol.

Malgrat això, però alhora donant més punts de fortalesa a IPFS, el futur proper planteja nous reptes en la distribució de dades, els quals són poc viables com s'ha comentat anteriorment per al protocol HTTP, per això es vol aconseguir solucionar-ho mitjançant nous protocols i sistemes com és el sistema distribuït IPFS.
Alguns dels reptes que es plantegen són els següents:

- Allotjament i distribució de grans conjunts de dades.

- Computació de moltes dades a través d'organitzacions - Big Data

- Contingut en temps real (Streaming) d'alta qualitat.

- Enllaçar conjunts de dades massius.

- Evitar la desaparició accidental d'arxius importants, entre altres reptes als quals IPFS vol donar solució.

## 1.4 Tecnologies relacionades

En el següent apartat s'expliquen els models i sistemes amb els quals s'ha basat IPFS i/o es basarà en un futur proper.

### 1.4.1 Taula de Hash Distribuïda - DHT

Una Taula de Hash distribuïda (àlies en anglès DHT - Distributed Hash Table) és una estructura de dades que s'utilitza en sistemes distribuïts per a indexar i buscar entre un nombre elevat de dades de les quals es necessita una cerca ràpida i eficient. Per a poder aconseguir-ho, es generen parelles <Clau-Valor>, on la clau, que és única per cada element de la taula, és l'element que s'utilitza per a buscar un determinat valor - en IPFS la clau és el hash criptogràfic de l'objecte mentre que el valor és l'objecte emmagatzemat.
El concepte distribuït s'identifica directament amb els nodes, els quals guarden les Claus (identificadors) en un sistema d'enrutament comú que permet trobar de forma eficient el node que té la informació d'interès.

La responsabilitat de mantenir el mapa de claus amb els valors corresponents està distribuït entre els nodes, de manera que un canvi en el conjunt causi la mínima interrupció de la xarxa i que els nodes siguin capaços de trobar el node responsable de una part de la taula per tal d'obtenir la informació necessària. Un dels avantatges que tenen les DHT és l'escalabilitat i la capacitat de combatre pèrdues de nodes, i com a conseqüència, de la informació que emmagatzema cada node.

En IPFS, els nodes utilitzen Kademlia DHT, explicat a l'apartat 1.4.2, per a trobar altres nodes que tenen un arxiu d'interès amb l'ajut de la clau, o sigui del Hash de l'arxiu.

Per tal de introduir i indexar a la xarxa totes les dades de la Taula de Hash de forma que sigui capaç de buscar, tot arxiu o dada es passa per una funció de Hash i s'obté una clau que quedarà identificada a la Taula de Hash dels nodes d'interès dins la xarxa distribuïda i permetrà als demés Peers/ nodes poder trobar l'arxiu en cas de necessitat.

Una vegada s'ha creada la DHT, els buckets que contenen els hash es distribueixen entre tots els nodes que hi ha a la xarxa i així es crea una xarxa de superposició connectada en enllaços lògics i establint l'esquema d'enrutament. El comportament de la DHT és el següent:

1. Un node coneix el hash d'un objecte i el vol trobar a la xarxa.

2. El node envia un missatge a la xarxa buscant el node responsable de l'objecte.

3. El node responsable respon amb l'objecte.

Les característiques més rellevants en la implementació de les DHT són l'autonomia i la descentralització, és a dir que els nodes són capaços d'organitzar-se de forma col·lectiva i crear un

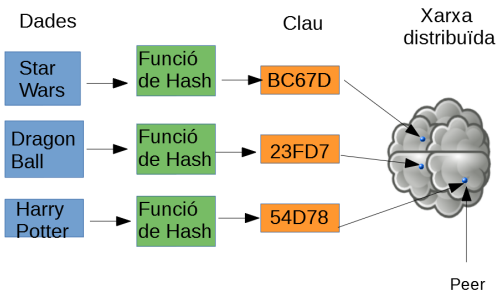


Figura 1.5: DHT - Taula de Hash distribuïda

sistema sense un òrgan central. El sistema és tolerant a errors, és a dir, ha de confiar en nodes que accedeixen i surten contínuament de la xarxa.

Les taules de hash distribuïdes són escalables i funcionen de manera eficient amb milers i milions de nodes.

Si ho avaluem a nivell matemàtic obtenim una relació de O(n) a O(log(n)) respecte al volum de dades a buscar utilitzant taules de hash distribuïdes i parelles <clau, valor>.

### 1.4.2 Kademlia DHT

Kademlia DHT [3] és un protocol per a implementar en les Taules de Hash distribuïdes, s'utilitza en sistemes Peer-to-Peer i és la DHT que s'utilitza actualment IPFS, tot i que la idea és millorar en un futur proper en sistemes més segurs com és S/Kademlia i Coral DSHT que s'expliquen en els següents apartats.

A diferència d'altres DHT, a Kademlia s'utilitza un sistema d'enrutament en forma d'arbre binari, on cada "fulla"de l'arbre és un node de la xarxa. Cada node manté contacte amb almenys un altre node de la seva branca. Cada branca està formada per un k-bucket.

En aquest arbre, cada node té una tripleta on es llista l'adreça IP, el port i el nodeID i parelles <clau, valor> per a intercanviar informació amb els de més nodes.

Els nodes, els arxius i les paraules clau, s'encripten en hash SHA1 en un espai de 160 bits. Cada node només conserva la informació dels nodes més propers de l'arbre.

Per a cada i entre 0 i 160 (0< i < 160), cada node manté una llista de nodes de distància entre $2^i$, i $2^{i+1}$ de si mateix on cada llista és un k-bucket. La llista classifica la informació respiecte l'última vegada que l'ha utilitzat amb nova informació.

El valor de "k"s'escull de manera que qualsevol conjunt de nodes "k"determinat no pugui perdre la informació de l'última hora (per defecte). La llista s'actualitza sempre que un node rep un

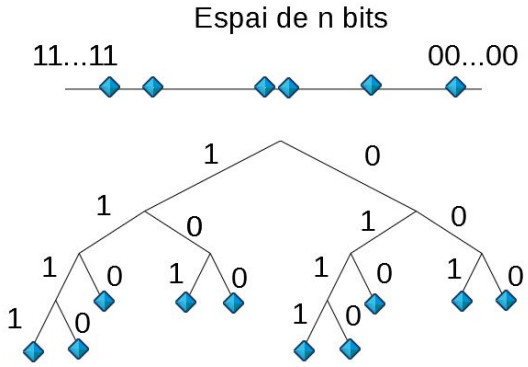


Figura 1.6: Arbre binari de Kademlia-DHT

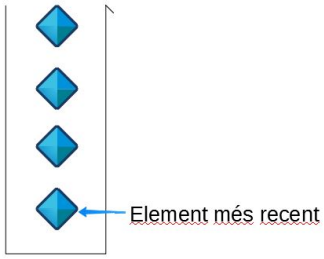


Figura 1.7: Figura on es mostra l'element més recent dins un k-Bucket

missatge.
Instruccions capaces de fer per Kademlia DHT:
- PING: Per tal de comprovar si un node està online tenim la possibilitat d'enviar un missatge Ping.

- Emmagatzemar: Instruccions a un node per emmagatzemar una clau.

- Find_Node: A partir de l'ID, el receptor retorna la tripleta amb l'adreça IP, el port UDP i el node ID dels "k"nodes que són més propers al node que estem buscant.

- Find_Value: Actua igual que l'instrucció Find_Node, excepte si el receptor emmagatzema la clau i retorna el valor guardat.
Per emmagatzemar una parella (clau, valor), un node localitza els k nodes més propers a la clau i els envia un missatge STORE RPCs indicant que els té emmagatzemats. Per a Kademlia, l'usuari que ha publicat el (clau, valor) es requereix per tornar a publicar-lo cada 24 hores. En cas contrari, les parelles caduquen 24 hores després de la seva publicació.

Hi ha una conjunt d'aplicacions que utilitzen Kademlia DHT, com són Gnutella i BitTorrent,

amb xarxes de més de 20 milions de nodes i que generen un interès per a la millora del protocol.

### 1.4.3 S/Kademlia

S/Kademlia [4] és una extensió de Kademlia que resolt alguns dels problemes a nivell de seguretat.

S/Kademlia aconsegueix que la creació de molts nodeId provoqui la necessitat de recursos, per evitar així el Sibil atac [8]. El Sibil atac consisteix en crear molts nodes per a controlar una fracció de la xarxa. En les xarxes centralitzades es pot fer pagar econòmicament als Peers per accedir a la xarxa i que sigui relacionat a un usuari, en canvi a una xarxa distribuïda la forma de pagament per tal que un usuari maliciós no creí molts IDs s'aconsegueix fent que necessiti molts recursos (ample de banda, diverses CPU 's, etc). Amb l'ajut de cripto puzles (CP) i criptografia asimètrica de clau pública (PKI), aconseguim evitar la creació de molts IDs, ja que cal xifrar la identificació dels nodes per identificar-se i enviar missatges signats.

Segons s'indica al paper de S/Kademlia [4], la forma més eficient de crear nodeId amb la màxima seguretat en xarxes sense una autoritat central que ens permeti autoritzar cert valor, són els cripta puzles. Existeixen diversos tipus de CP, però els següents esquemes mostren els CP estàtics i dinàmics, els quals impedeixen que l'usuari pugui triar lliurement el nodeId que se li assignarà i el dinàmic mostra una complexitat en la creació de molts nodeId.

Els nodes de S/Kademlia creen rutes diferents per a trobar un cert contingut per tal de garantir que els nodes "bons" obtinguin el contingut i evitar així que Peers "maliciosos" difereixin en la cerca de contingut. S/Kademlia proporciona una taxa d'èxit per a trobar el contingut d'un 85%.

### 1.4.4 Coral DSHT

Coral DSHT [5] és una extensió de la taula de hash (DHT) utilitzada per Kademlia anomenada Distributed Sloopy Hash Tablem (DSHT) que permet disminuir l'ús d'ample de banda del sistema o l'emmagatzematge de contingut on no és necessari i evitar així que els nodes que tenen arxius "populars", i que molts nodes demanen siguin punts Hotspot o punts excessivament carregats amb l'ajut de les DSHT.

Les DSHT creen clústers de nodes que s'organitzen de forma automàtica i obtenen informació entre ells per evitar comunicar-se amb servidors més llunyans o nodes carregats.

El concepte Taula de Hash Sloopy fa referència al fet que Coral està format per diversos "anells"de Taules Distribuïdes, on cada anell representa un nivell de localització dels nodes en funció del temps de ping. Cada DHT està formada per nodes que es troben dins una latència entre ells, per exemple un anell de nodes que es troben a 20 mil·lisegons entre ells.
Coral millora Kademlia en diversos aspectes:

Mentre que les DHT tradicionals proporcionen dues funcions, una per introduir valors a una clau especifica (put(key, value)) i una per obtenir un valor a partir de una clau (get(key)), el protocol de Coral és capaç de crear diversos valors adreçats a una clau. En les DHT tots els nodes que demanin una mateixa clau contactaran sempre amb el node més proper a la clau i carregaran excesivament el node, conegut com un overloaded node o un hotspot node.
Quan utilitzem la funció get(key) en Coral DSHT els nodes són capaços de guardar el contingut, avisar als nodes veins que tenen el contingut (Have-list) i evitar que un contingut concret l'hagi d'anar a buscar directament a l'origen i el podem tenir guardat al nostre node. És a dir, quan busquem un contingut pel seu Hash, ens acostem al valor del Hash a partir dels valors dels NodeId semblants al Hash (find-closer-node(key)) els quals tenen una probabilitat molt alta de tenir el contingut o tenir identificat el Node que té el contingut.

Tot comença quan l'origen puja el Hash del contingut sobre el nodeId més semblant al Hash del contingut indicant que el seu Hash és el que emmagatzema el valor. Per a trobar el node més semblant al contingut ens hem anat "acostant" al node a partir de nodes semblants al nostre Hash fins a trobar el més semblant i alhora informant a aquests nodes que el nostre node guarda l'arxiu/valor.

Un cop el node més semblant al hash de l'objecte guarda el nostre id <hash de l'objecte (igual o molt semblant al node que redirecciona), nodeId meu> és capaç de indicar als demés nodes que necessitin l'objecte com trobar-lo. Si un node demana l'arxiu i se'l descarrega, és capaç de indicar a els nodes propers al hash de l'objecte que ell també és capaç de donar l'arxiu (ara tenim el node creador de l'arxiu i un altre node capaç de enviar el contingut), així no creem un hotspot sobre l'origen creador de un arxiu si és molt popular, com podria ser el document IPFS P2P de Juan Benet, i aconseguim trobar el contingut de forma ràpida i eficient. I els nodes propers al contingut també alberguen la redirecció per a trobar el contingut.

Una altre millora de Coral respecte Kademlia és que Kademlia guarda en una taula els nodeId 's més propers respecte la distància XOR ignorant si aquests nodes necessiten la informació que tu tens i alhora ignora nodes llunyans els quals tenen interès del contingut que els pots proporcionar. Coral guarda adreces de nodes que poden proporcionar-nos contingut d'interès.

A més, Coral s'organitza en format jeràrquic utilitzant Clústers separats en funció del temps per a trobar un altre node. Així els usuaris poden buscar primer en el Clúster més proper a ells mateixos (el qual tindrà un RTT menor) i reduir la latència de la cerca si en aquest i troben la informació d'interès. En cas de no trobar la informació en el clúster més proper, es segueix la cerca en el segon clúster fins a trobar la clau més propera a la que es busca.

També es important remarcar que quan demanem un hash a la xarxa i un dels Peers ens l'envia, al rebre'l es comprova si és el hash que hem demanat, que és únic (ja que està adreçat a contingut) i no és creen copies a la xarxa de un mateix arxiu.

### 1.4.5 Merkle Direct Acyclic Graph (DAG)

Al nucli de IPFS hi trobem Merkle DAG (Direct Acyclic Graph) com també ho és en la tecnologia GIT. El contingut es representa en forma d'arbre i és identificat per el seu propi hash criptogràfic.

IPFS és una mica més complex una vegada obtenim el hash de l'objecte codificat ja que tot arxiu que té un volum superior a 256kB es divideix en blocs d'aquest tamany.

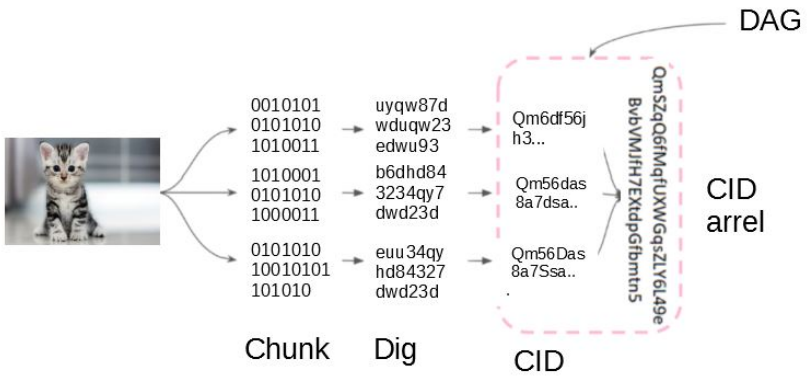


Figura 1.8: Arxiu amb un volum superior a 256kb subdividit i organitzat en un Merkle DAG

Els arxius que tenen un tamany superior a 256kB es subdivideix tants blocs d'aquest tamany com sigui necessari, és a dir si afegim un arxiu de 1.08Mb a la xarxa IPFS amb la comanda ipfs add arxiu, es dividirà en 5 arxius i hashos diferents, 4 d'ells de 265kb i l'últim bloc de 84kb i cada un d'ells estarà identificat pel seu Hash. És a dir, el nostre arxiu ha quedat trossejat en 5 blocs i s'han organitzat en una estructura d'arbre i enllaçats permeten obtenir de nou el hash arrel de l'objecte.

Exemple:

Importem al nostre node una imatge de 1 MB, n'obtenim el hash arrel del contingut i amb la comanda ipfs ls -v observem que s'ha dividit en 5 blocs.

```
>ipfs add planet.jpg
1.08 MiB / 1.08 MiB [==============================
==================] 100.00%added
 QmSZqQ6fMqfUXWGqsZLY6L49eBvbVMJfH7EXtdpGfbmtn5 planet.jpg
1.08 MiB / 1.08 MiB [==========================
==================] 100.00%

>ipfs ls -v QmSZqQ6fMqfUXWGqsZLY6L49eBvbVMJfH7EXtdpGfbmtn5
Hash                                           Size   Name
QmWTmuguujYEeJ7yNtGr78bM7ogP1vKWybveEF3ZuUeBmi 262158
QmdV9iDrji8RSuJke43si3o2PLGL1fQoBRq1tP87HVDhEq 262158
QmYqcLhqCZKq39X2cvwG5TBmdy9MUYNHcYpCfvJzc11m1q 262158
QmZ8WHnhrPWXYWG8PFSDG5hXPPdKeR7bShCxC3P1mUo2pn 262158
```

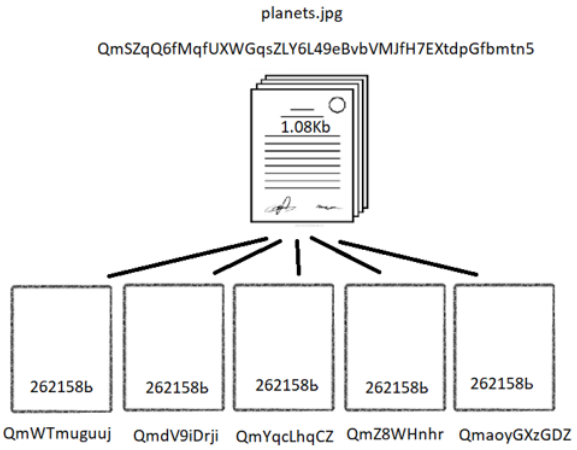


Figura 1.9: Fitxer subdividit en diversos blocs

```
QmaoyGXzGDZ6Uq7thtmFciDacT6U85szwTMjzNL58mSFk2 84379
```

Podem observar que el nostre Hash de planet.jpg està representat per una llista de links dels blocs d'arxius inferiors a 256kB que especifiquen que el nostre arxiu/objecte és un fitxer gran (>256kB).

```
>ipfs object get QmSZqQ6fMqfUXWGqsZLY6L49eBvbVMJfH7EXtdpGfbmtn5
{"Links":[{"Name":"","Hash":"QmWTmuguujYEeJ7yNtGr78bM7ogP1vKWybveEF3ZuUe
Bmi",
"Size":262158},{"Name":"","Hash":"QmdV9iDrji8RSuJke43si3o2PLGL1fQoBRq1tP87HVD
hEq",
"Size":262158},{"Name":"","Hash":"QmYqcLhqCZKq39X2cvwG5TBmdy9MUYNHcYpCfvJzc11
m1q",
"Size":262158},{"Name":"","Hash":"QmZ8WHnhrPWXYWG8PFSDG5hXPPdKeR7bShCxC3P1mUo
2pn",
"Size":262158},{"Name":"","Hash":"QmaoyGXzGDZ6Uq7thtmFciDacT6U85szwTMjzNL58mS
Fk2",
"Size":84379}],"Data":"\u0008\u0002\u0018\ufffd\ufffdE \ufffd\ufffd\u0010
ufffd\u0010 \ufffd\uff
fd\u0010 \ufffd\ufffd\u0010 \ufffd\ufffd\u0005"}
```

Aquest comportament en els objectes ens proporciona propietats molt útils i interessants l'IPFS:

- Autenticat: el contingut està representat per un hash i permet verificar els links.

- Permanència: una vegada que s'obtenen l'objecte es pot emmagatzemar en memòria.

- Universal: qualsevol estructura de dades es pot representar com un Merkle DAG.

- Descentralitzat: qualsevol usuari pot crear objectes, sense necessitat de un servidor central.

A l'hora, aquestes propietats també permeten l'intercanvi de fitxers entre usuaris que no es co-

neixen (però estan autenticats a al xarxa), els objectes es poden utilitzar offline i a més qualsevol estructura de dades es pot importar a la xarxa IPFS.

### Estructuració directoris

I com s'estructuren els directoris? Els directoris s'estructuren de forma lògica seguint les carpetes, subcarpetes i arxius de cada directori amb la diferència que tot arxiu que estigui duplicat en ens directoris tan sols s'emmagatzemarà en una instància i podrà ser enllaçat/linkat pel seu Hash respecte altres directoris.
Creem una carpeta Directori i hi introduïm un arxiu amb el contingut "Hello World" i una subcar-

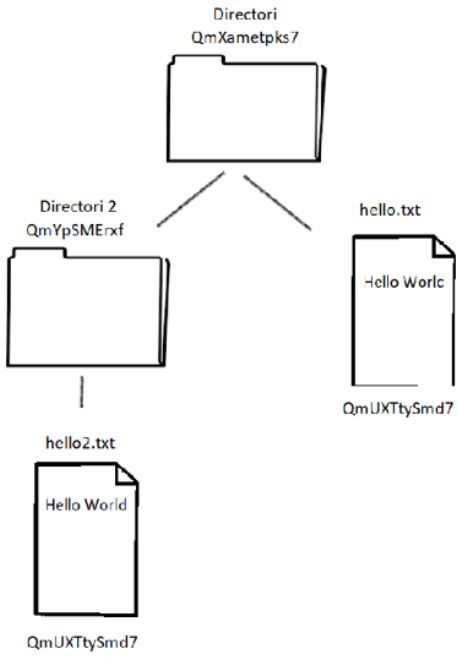


Figura 1.10: Directori amb diversos fitxers

peta amb un arxiu amb el mateix contingut "Hello World".

Podem accedir als arxius amb el Hash establert del contingut (ipfs/H(Directori)/Directori2/hello2.txt) o amb el Path desde el hash del directori (ipfs/H(hello2.txt) )

Quan visualitzem l'objecte, observem que només s'han introduït dos objectes al Directori, ja que com s'ha indicat al llarg del document els hashos estan adreçats a contingut i el contingut no és duplica a la xarxa.

En el cas que creem un nou directori a la nostra xarxa que tingui algun dels fitxers que ja han estat introduïts a la xarxa anteriorment no caldrà que el repositori els guardi, sinó que tan sols quedaran referenciats pel hash del seu link.

### 1.4.6 Bitswap

L'intercanvi de blocs en IPFS entre Peers o nodes utilitza un model inspirat en BitTorrent. El protocol Bitswap permet demanar els blocs i enviar-los a altres nodes de la xarxa (enrutat gràcies a les DHT) i a més s'encarrega de enviar els blocs als nodes que els necessiten de forma estratègica.

La versió que utilitza IPFS està en procés de millora, però tot i això treballa de forma eficient. Bitswap actua com un protocol basat en missatges, on tots els missatges contenen una Want-List o blocs de dades desitjats.
Els nodes tenen 2 tipus de llistes, la d'elements desitjats que demana als demés, o Want-Lis i la de objectes que té, àlies Have List.

Quan un node rep una llista Want-List, provinent de un altre node, comprova si té algun element de la llista i en cas de tenir-lo, l'envia al node emissor. Una vegada un node ha rebut el bloc d'interès, envia una notificació indicant que aquell bloc ja el té i no l'hi envïin de nou.
L'acció d'enviar la Want-List s'envien quan s'obra connexió, de forma periòdica, quan es produeix un canvi en la Want-List o quan ha rebut un bloc nou.

Quan un node rep una Want-list de un altre node, la guarda i comprova si té algun dels blocs que necessita l'altre node. Si en té, utilitza la Bitswap Strategy. Un cop rebut fa una comprovació de que el hash rebut és el hash que esperava, canvia l'element de la Want-List a Have-List.
Es tanca la connexió quan expira un Timeout de 30 segons sense rebre cap informació.

## 1.5 IPFS en profunditat

### 1.5.1 Fingerprinting

De forma semblant per a identificar un llibre amb els valors de ISBN, en IPFS necessitem una forma de donar valor als continguts mitjançat l'adreçament de contingut.

En la pràctica, l'adreçament de contingut en IPFS es fa mitjançant hashos criptogràfics únics per a cada objecte. Per tal de aconseguir-ho, posem l'arxiu, imatge, contingut, etc. en format binari, fem la funció de hash al valor binari, així assegurem que el contingut serà únic per a cada arxiu. Si modifiquem un dels valors binaris de l'arxiu, la funció de hash produirà un valor totalment diferent a l'anterior.

Una vegada tenim la funció de hash, és necessari convertir-ho en quelcom que IPFS sigui capaç de localitzar i "entendre".

Si ens fixem en el Tutorial de l'apartat 6, tots els hashos tenen 34 bytes i comencen per Qm... (base58 a hexa és 0x1220), que en format hexadecimal signifiquen 0x12 - sha2 (hash algorithm) i 0x20 (mida del contingut) - 256 bits de longitud, conegut com a sha256-256 per a la comunitat de IPFS. És a dir, tenim una funció de hash formada per a diferents conceptes i no sols per al hash del contingut, aquest protocol s'anomena Multihash.

### 1.5.2 MultiHash

Un Multihash és un protocol que permet que en el propi hash s'identifiqui l'algorisme que s'utilitza i la mida del hash resultant en només els dos primers Bytes del multihash. IPFS mostra el hash resultant en format base58 perquè elimina valors alfanumèrics semblants visualment, com el zero (0) i la o majúscula (O), o també la lletra el·la (l) i la lletra i majúscula (I) i així s'eviten errors humans alhora de llegir els hashos.
Amb el multihash podem permetre implementar nous mecanismes de Fingerprinting en el hash i que convisquin amb els actuals. És a dir, si es crea una nova funció de hash, molt més segura que l'actual, solament canviant els dos primers Bytes podrem permetre la implementació en IPFS i variarà el Qm... inicial actual.

### 1.5.3 Verificació

Quan el nostre node d'iguals de IPFS està comunicant-se amb altres nodes de la xarxa, passen diverses coses darrere de l'escenari. Una de les coses més importants és que els usuaris s'identifiquen entre ells a través del nodeID, el qual és el hash criptogràfic de la clau pública de l'usuari que es crea quan iniciem per primera vegada el protocol IPFS. Aquesta identificació proporciona una identitat única que permet verificar l'autenticitat pròpia i dels demés usuaris de la xarxa, i

d'aquesta forma els Peers/nodes saben que estan comunicant-se amb el Peer adequat. Per tal de efectuar la verificació, els dos nodes intercanvien les claus públiques i comproven que aquestes es corresponen al nodeID corresponent.

En el cas d'IPFS, el Hash criptogràfic és un multishash sha-256 com s'ha comentat anteriorment i representat en base58.

### 1.5.4 Routing - DHT

Actualment, l'enrutament en IPFS està basat en les Taules de Hash Distribuïdes de Kademlia DHT, tot i que com s'ha explicat en la secció 1.4 en un futur proper es vol migrar a Coral i S/Kademlia, que com s'ha indicat, estan basats també en Taules de Hash Distribuïdes.

### 1.5.5 Pinning

La pèrdua d'informació és un esdeveniment real i comú a l'internet actual degut a que si pugem informació en un servidor i aquest és formatejat, substituït o degut a una causa aliena, perdem la informació que hi teníem guardada. En IPFS, tenim la possibilitat de "pinnejar" el contingut i que el nostre node local no perdi aquesta informació.

Per a controlar i no eliminar/perdre contingut fem pin al hash d'interès i mantindrem l'arxiu al nostre repositori.
Com funciona i per a què és necessari fer Pin a contingut que no ens interessa perdre a la xarxa IPFS?
IPFS té la opció d'activar un Garbage Collector (GC) quan iniciem el Daemon, que com el mateix nom indica el que fa és eliminar el contingut brossa del nostre node. I quin contingut elimina? Doncs per defecte el GC elimina tot el contingut que no ha estat Pinnejat del nostre repositori, i té un temps per defecte de una hora (configurable) com es mostra en el informació configurable del GC:

```
>ipfs config Datastore
{
        "BloomFilterSize": 0,
        "GCPeriod": "1h",
        "HashOnRead": false,
        "Spec": {
                "mounts": [
                {
                        "child": {
                                "path": "blocks",
                                "shardFunc": "/repo/flatfs/shard/v1
                                "sync": true,
```

```
                                "type": "flatfs"
                        },
                        "mountpoint": "/blocks",
                        "prefix": "flatfs.datastore",
                        "type": "measure"
                },
                {
                        "child": {
                                "compression": "none",
                                "path": "datastore",
                                "type": "levelds"
                        },
                        "mountpoint": "/",
                        "prefix": "leveldb.datastore",
                        "type": "measure"
                }
                ],
                "type": "mount"
        },
        "StorageGCWatermark": 90,
        "StorageMax": "10GB"
}
```


Cal remarcar que quan afegim un contingut amb la comanda ipfs add çontingut", simultàniament queda el hash del contingut que es genera "pinnejat".

El contingut es mantindrà al nostre node sempre que no eliminem el pin del hash o bé el node no sigui eliminat. Altrament, el contingut serà mantingut al nostre node i podrà ser accedit per altres peers quan el nostre node estigui a la xarxa.

Observem doncs que el concepte de permanència en la xarxa de IPFS s'identifica des de un punt de vista local, ja que per molt que un altre node hagi Pinnejat un contingut, si per algun motiu, com la pèrdua de connexió a internet, no som capaços d'arribar a l'altre node o aquest ha estat eliminat, el seu contingut "pinnejat"serà inexistent per a nosaltres i no podrem descarregar-lo ni identificar-lo com a permanent a la xarxa.

Per a mostrar com funciona el pin a l'apartat 1.6.6 es mostra un petit tutorial en el qual s'introdueix un contingut des del node, visualitzarem el contingut, s'eliminarà el pin del hash del contingut, s'activarà de forma manual el Garbage Collector (que eliminarà tot contingut del repositori que no hagi estat pinnejat) i observarem que no es pot visualitzar si no és que un altre Peer ha descarregat el contingut o té en el seu repositori el fitxer per casualitat.

Ens adonarem doncs que si aquest fitxer no l'ha descarregat/introduït a la xarxa cap més Peer al que nosaltres podem arribar, no serà possible visualitzar-lo. Per exemple si eliminem un arxiu amb contingut "Hello World", que possiblement diversos Peers de la xarxa l'han creat en el seu inici, tot i eliminar-lo del nostre repositori, la xarxa serà capaç de mostrar-nos aquest contingut.

### 1.5.6 IPNS - Inter Planterary Name System

Observem que amb IPFS cada vegada que modifiquem el contingut obtenim un hash nou degut a la immutabilitat que aporta el sistema. Si tenim una web la qual el contingut no varia mai, no tindrem cap problema ja que sols redirigim el nostre DNS a on tenim ubicat els arxius IPFS, però si tenim un web molt més dinàmica, i l'acutalitzem de forma contínua, sembla que IPFS no ens ajuda molt ja que cada contingut genera un nou hash com s'ha comentat anteriorment i es visualitzarà al tutorial.

Com ho aconsegueix IPFS? Doncs re-adreça el hash del contingut al nostre hash del node ID.

La idea de IPNS és trencar un dels principis de immutabilitat de IPFS, per a què els nodes d'interès estiguin al corrent del contingut més actualitzat. Permet emmagatzemar la referència d'un hash ipfs i actualitzar-ho amb una ordre de publicació.

```
ipfs name publish  HASH NOU.
```

L'únic que cal fer cada cop que modifiquem el contingut d'una web, és posar el nou hash i publicar-l'ho, sense la necessitat d'entrar al DNS o canviar el host, es tracte de fer un "redeploy" de pocs segons i així actualitzar els fitxers amb l'últim contingut.

### 1.5.7 Path

IPFS permet obtenir els fitxers existents al sistema de diverses formes, directament a partir del hash de l'objecte afegint /ipfs/ al davant, és a dir:

ipfs/H(FOO.TXT), o bé a partir del directori pare del fitxer també el podem obtenir de la forma /ipfs/H/FOO.TXT, o sigui tots els objectes son accessibles mitjançant el seu hash o bé el hash del directori que els emmagatzema.
Objectes locals És necessari que els usuaris que utilitzen IPFS tinguin un emmagatzematge local per a guardar els objectes que volen emmagatzemar a la xarxa.
El tipus de storage o magatzem depèn de cada usuari, però majoritàriament és una porció del disc dur o del Ram del PC.
Quan demanem un objecte a un altre Peer, el descarreguem i es guarda localment al nostre repositori.

### 1.5.8 PubSub

El protocol Publicador - Subscriptor, o conegut pels internautes com a sistema PubSub, és un protocol que es permet utilitzar en IPFS en el qual els Publicadors creen tòpics en els quals i poden posar contingut i els Subscriptors poden rebre-la subscrivint-se als tòpics d'interès sense la necessitat de tenir una connexió directa. Aquest sistema permet una millor flexibilitat i escalabilitat a la xarxa.

Algunes aplicacions que utilitzen el sistema PubSub estan relacionades per exemple en xats, documents per a treballar de forma col·laborativa (semblant al Google Docs), jocs multijugador, entre d'altres aplicacions P2P.

A l'apartat 1.6.8 s'explica en detall com utilitzar-ho des de el punt de vista del Publicador i del Subscriptor.

### 1.5.9 Seguretat - Encriptar fitxers pujats a la web

Observem que si qualsevol persona coneix el Hash de un arxiu que hem pujat a la xarxa IPFS, és capaç de descarregar-lo. Per tal de crear un arxiu "segur"a la xarxa i que sols un usuari concret sigui capaç de poder desxifrar-lo, la idea més simple i segura és posar una clau privada a l'arxiu que concordi amb el nodeID de l'usuari destinatari perquè sols ell sigui capaç de desxifrar un cop descarregat i així qualsevol intrús podrà descarregar l'arxiu però a l'obrir-lo no podrà desencriptar-lo.

## 1.6 Tutorials - Learning by doing

### 1.6.1 IPFS desde zero

Per tal de poder utilitzar el sistema de fitxers IPFS és necessari descarregar la versió més actualitzada de la web https://docs.ipfs.io/introduction/install/.

Un cop descarregat, accedirem a través del terminal a la carpeta on s'hagi creat el directori go-ipfs i iniciarem IPFS utilitzant la comanda ipfs init.

```
> ipfs init
initializing ipfs node at /Users/jbenet/.go-ipfs
generating 2048-bit RSA keypair...done
peer identity: Qmcpo2iLBikrdf1d6QU6vXuNb6P7hwrbNPW9kLAH8eG67z
to get started, enter:

ipfs cat /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv/readme
```

Seguint les instruccions del terminal, copiarem la ruta que ens indica i si s'ha instal·lat de forma correcta obtindrem la següent resposta:

```
Hello and Welcome to IPFS!

IPFS

If you're seeing this, you have successfully installed
IPFS and are now interfacing with the ipfs merkledag!

-------------------------------------------------------
| Warning:                                            |
|   This is alpha software. use at your own discretion! |
|   Much is missing or lacking polish. There are bugs.  |
|   Not yet secure. Read the security notes for more.   |
-------------------------------------------------------

Check out some of the other files in this directory:

./about
./help
./quick-start     <-- usage examples
./readme          <-- this file
./security-notes
```

Una vegada completats els passos anteriors, s'ens ha creat un NodeID personal únic i un repositori on guardar els nostres fitxers de forma local.

Per a conèixer el nostre nodeID, el qual ha estat format per la funció de Hash sobre la nostra clau pública, podem saber-lo a través de la següent comanda:

```
>ipfs id

{
        "ID": "QmTrYg8LBa7WwgCp6SqDdH3xHk2Z4iCKC6zCKJuNyLjuLQ",
        "PublicKey": "CAASpgIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDV
        +Bqvzx6eWR4s6
        vtwfcw910WNEsV5Nrtze8ah3HKfD8VtFPsai3sOXSpr2aXsHSVyM0Fx9envjYqaaa8v
        tG/4IwgZXc/sKXBE9mtad9552mjVhOqlzm+25XG4to7C/Plw6cgN6ZDxTn/Evi82mCh
        wmHl1qsFgJ9kYMfNqb3jvFdVU4HC02ZUmhSYOgHbhfSkpqW6pRq++IX77ui/u09sp1r
        veGFATuL3gSsvO7I2TTnfv+qIjDeON2Lou1EDM9dm4G1pNV6o46OuVHsuta5kq140ac
        1odyIzW3QrJE2VVsE0oaPbgTzMHUftznvTnXKPsLRNAq5kERDuF7z28NT+lAgMBAAE=",
        "Addresses": null,
        "AgentVersion": "go-ipfs/0.4.17/",
        "ProtocolVersion": "ipfs/0.1.0"
}
```

A més, també podem pujar arxius al nostre repositori local amb la comanda add. A la resposta del terminal observarem que es crea un Hash únic relacionat directament amb el contingut de l'arxiu:

```
>ipfs add "Prova.docx"
515.24 KiB / 515.24 KiB [======================
======================] 100.00%
added QmUU6TRvRwd8E3TzaBDky7NLcyJXcuGPeahM5BwqBZ7cD6 Prova.docx
515.24 KiB / 515.24 KiB [======================
======================] 100.00%
```

També tenim la possibilitat d'introduir una carpeta que contingui diversos arxius, per a aconseguir-ho hem d'afegir la comanda -r add CARPETA i ens retornarà el hash de tots els objectes que formen part del directori, a més del propi hash del directori que ens permetrà accedir als arxius de forma directe utilitzant el Hash de cada objecte o bé a través del hash del directori + path de l'arxiu respecte el directori "pare".

```
>ipfs add -r Carpeta
85.53 KiB / 1.25 MiB [=====>---------------
-------------------------]   6.70%
added QmZTfzTjSexZKhDRNw8PzvQkdCNEzWcQAV
GGpEMX7CgLXL Planets/IEEE\_IPFS.pdf
1.16 MiB / 1.25 MiB [======================
====================>-----]  93.30%
added QmSZqQ6fMqfUXWGqsZLY6L49eBvbVMJfH7EXtdpGfbmtn5 Planets/planet.jpg
1.25 MiB / 1.25 MiB [======================
========================] 100.00%
```

### 1.6.2 Path

Com s'ha comentat a l'apartat 1.5.7, podem obtenir un fitxer mitjançant la ruta completa o bé a directament a partir del hash de l'objecte:

En aquest exemple tenim la següent estructura de carpetes i l'afegim al nostre repositori:


Figura 1.11: Exemple - Estructura de carpetes

```
>ipfs add -r D1
2 B / 2 B [=============================================
==============================================] 100.00%added
 QmakYENTot78QdDQABVGxxem8iyF3p4qodSBPiXp1
n32hj D1/D2/F1.txt
2 B / 2 B [=============================================
==============================================] 100.00%added
 QmT5ZgddzsP964BLhUmtHYLJqQJKVjgxabRFY3mQU
6bqkJ D1/D2
2 B / 2 B [=============================================
==============================================] 100.00%added
 QmUWzs3vyeHCQgNtHVeksYG4XNHASdmAmWDNneJtW
WzfdW D1
2 B / 2 B [=============================================
==============================================] 100.00%
```

Podem obtenir el mateix fitxer a través del Hash directe o bé utilitzant la ruta completa a partir del Hash del directori arrel:

```
>ipfs get QmakYENTot78QdDQABVGxxem8iyF3p4qodSBPiXp1n32hj
Saving file(s) to QmakYENTot78QdDQABVGxxem8iyF3p4qodSBPiXp1n32hj
10 B / 10 B [=============================================
==============================================] 100.00%

>ipfs get QmUWzs3vyeHCQgNtHVeksYG4XNHASdmAmWDNneJtWWzfdW/D2/F1.txt
Saving file(s) to F1.txt
10 B / 10 B [=============================================
==============================================] 100.00%
```

### 1.6.3 Online - Daemon

Per tal de tenir un Daemon o servidor local connectat a la xarxa IPFS i ser capaços de obtenir fitxers de altres nodes hem d'activar en un nou terminal la següent comanda, la qual inicialitza tots els ports de IPFS fins que el Dameon està a punt i estem ONLINE!:

```
C:\Users\jordi\Desktop\go-ipfs>ipfs daemon
Initializing daemon...
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/169.254.128.121/tcp/4001
Swarm listening on /ip4/169.254.229.236/tcp/4001
Swarm listening on /ip4/192.168.1.39/tcp/4001
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /p2p-circuit/ipfs/QmTrYg8LBa7WwgCp6SqDdH3xH
k2Z4iCKC6zCKJuNyLjuLQ
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/169.254.128.121/tcp/4001
Swarm announcing /ip4/169.254.229.236/tcp/4001
Swarm announcing /ip4/192.168.1.39/tcp/4001
Swarm announcing /ip6/::1/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

### 1.6.4 Get i Cat

Una vegada estem connectats amb el daemon a la xarxa IPFS som capaços de obtenir fitxers d'altres nodes. Per a fer-ho tenim les comandes Get, la qual ens permet descarregar l'objecte, i la comanda Cat que ens permet visualitzar-lo en la pantalla.

En el següent exemple buscarem un arxiu comú que té el contingut Hello World, que està identificat amb el hash: QmXcMeYV3JZpdJCSDy2CGpEGdJAWW7qQKdnjuSPmpgeWZE.

Amb la comanda get podem descarregar el fitxer:

```
>ipfs get QmXcMeYV3JZpdJCSDy2CGpEGdJAWW7qQKdnjuSPmpgeWZE
Saving file(s) to QmXcMeYV3JZpdJCSDy2CGpEGdJAWW7qQKdnjuSPmpgeWZE
22 B / 22 B [=================================================
=======================================] 100.00% 0s
```

Amb la comanda Cat, visualitzarem el contingut del fitxer:

```
>ipfs cat QmXcMeYV3JZpdJCSDy2CGpEGdJAWW7qQKdnjuSPmpgeWZE
Hello World
```

### 1.6.5 DHT

Comanda per a trobar els nodes de la nostra xarxa que ens proporcionen un contingut específic, per exemple un text on diu “hello”:

1. Insertem a la xarxa de ipfs el text “hello” i n’obtenim el hash.

2. Confirmem que el contingut del hash és el text “hello”.

3. Amb la comanda ipfs dht findprovs hash del contingut obtenim els nodes que ens poden proporcionar el contingut amb el hash específic

```
>echo 'hello' | ipfs add
10 B / ? [-------------------------------------------------------=-
--------] added QmQTZisGUwXdHHEAuc967Qzu7QrjLAf8dcC1jyh1MwXuh6
QmQTZisGUwXdHHEAuc967Qzu7QrjLAf8dcC1jyh1MwXuh6
10 B / ? [-------------------------------------------------------=---------]

>ipfs dht findprovs QmQTZisGUwXdHHEAuc967Qzu7QrjLAf8dcC1jyh1MwXuh6
QmTrYg8LBa7WwgCp6SqDdH3xHk2Z4iCKC6zCKJuNyLjuLQ

>ipfs dht findprovs QmQTZisGUwXdHHEAuc967Qzu7QrjLAf8dcC1jyh1MwXuh6
QmLBa7WwgCp6SqDdLBaRw432Mtgk2Z4iCKC6zg8LBa7dLB
QmTrYg8LBa7WwgCp6SqDdH3xHk2Z4iCKC6zCKJuNyLjuLQ
QmNnMbBrRw432Mtg8LBa7WwgCp6SqDdLBa7WwHk2Z4iCKC
Qmp6SqDdH3xHk2Z4iCg8LBa7WaRw432Mtgk6SqD2Z4iCuj
...
```

Per obtenir l’adreça de qualsevol dels nodes, per exemple del primer, tan sols cal utilitzar la següent comanda:

```
>ipfs dht findpeer QmLBa7WwgCp6SqDdLBaRw432Mtgk2Z4iCKC6zg8LBa7dLB
/ip4/9.0.7.210/tcp/4001
/ip6/::1/tcp/4001
/ip4/52.209.239.213/tcp/4001
```

Podem conèixer els nodes propers a un Peer amb la següent comanda, on el hash és el nodeID:

```
>ipfs dht query Qmf5gJSJ8wNBQanfBNh2AP5qyPuXVrnt9zWtrgqhbtuZhb
QmRNHaxHqnFKZdLKSP1wpuZ1F1MQ4qvTp42Em55ENUuAhq
QmeUGfvXGstd5fVRpCjGXnqdYb5VK8oxQn9YqadUhHMapr
QmRPMTTbuZhPGRu7baDxtKrWUHxy5PRxHffz9NyNDX2p1m
QmeBdH3EPwuMptPY6MygVeABqCPkFwS9SWo91wrjDJTEKy
Qma6ojAwcMx4rj3CQ7yCjhXTXVu12A1zbMnMJmKDDoxTeA
QmePbw55u2Vto1MAAWHUUEFe9S7dwNWcfBDBQRpRWgP4gX
QmQT7HCFy8yAmT4VaEEL3jpwwXvtpZqxna6XrajmWNHi3B
QmbinWcsuo13DUMtKGKNtXR3yEBrNb8CQtiFuyzZdvGwLx
QmXzB5LWXQFx1ogFX8vxP8bMUCWSfiVfsft51sTUop7M7M
QmTiCerJDqrC6vLDgcNzu8qNK4e9rTx6bJLtSgQHpkUUfk
QmcxKn3WEwWTf5WCnWD9WPnq4mfu1zXkMDgVKCNFJ6NpD
```

```
...
```

### 1.6.6 Pinning i ús del Garbage Collector

A continuació hi ha una petita pràctica per a entendre el Pinning i el Garbage Collector en IPFS. Per defecte quan afegim un arxiu ( ipfs add arxiu) al sistema IPFS es Pinneja per defecte, i coma recordatori, quan Pinnejem un objecte evitem que si s'activa el Garbage Collector - Eliminador de brossa/ arxius no pinnejats s'elimini el contingut del nostre repositori.

En el següent exemple s'introdueix el text "Telecos"al nostre repositori i observem que queda hashejat amb la comanda cat que ens retorna el valor del Hash:

```
>echo Telecos | ipfs add
10 B / ? [-----------------------------------------=---------] added
 QmZmUGsBkXd87rydosUwLJ1yWqNDbKXAN9pJWsjTwFRM5r
10 B / ? [-----------------------------------------=---------]
s>ipfs cat QmZmUGsBkXd87rydosUwLJ1yWqNDbKXAN9pJWsjTwFRM5r
Telecos
```

A continuació, treiem el Pin de l'arxiu amb la comanda pin rm arxiu:

```
>ipfs pin rm QmZmUGsBkXd87rydosUwLJ1yWqNDbKXAN9pJWsjTwFRM5r
unpinned QmZmUGsBkXd87rydosUwLJ1yWqNDbKXAN9pJWsjTwFRM5r
```

En aquest punt, si no s'activa de forma automàtica el GC, l'arxiu es manté al repositori com podem observar:

```
>ipfs cat QmZmUGsBkXd87rydosUwLJ1yWqNDbKXAN9pJWsjTwFRM5r
Telecos
```

Però en el moment en què activem de forma manual el GC, l'arxiu s'elimina del nostre repositori i l'intenta buscar a la web. En cas que un altre node tingués per casualitat el text "Telecos"seriem capaços d'obtenir-lo. Com observem, el node es queda buscant l'arxiu no existent a la xarxa:

```
>ipfs repo gc
removed QmZmUGsBkXd87rydosUwLJ1yWqNDbKXAN9pJWsjTwFRM5r

>ipfs cat QmZmUGsBkXd87rydosUwLJ1yWqNDbKXAN9pJWsjTwFRM5r
```

### 1.6.7 Bitswap

Per a conèixer tots els blocs enviats, si tenim blocs pendent de la WantList i els companys que tenim identificats podem utilitzar la següent comanda:

```
 >ipfs bitswap stat
 bitswap status
 provides buffer: 0 / 256
 blocks received: 0
```

```
blocks sent: 7
data received: 0
data sent: 6189
dup blocks received: 0
dup data received: 0 B
wantlist [0 keys]
partners [775]
QmNMTsUaKTDdunSUN44TZuMPDpGTCLG8R2ooEo1ZmxJadq
QmNRSBnfjE6PMZ5AfnLccWpMjX2LtRFYedxQ2UqQxvozfV
QmNSYxZAiJHeLdkBg38roksAR9So7Y5eojks1yjEcUtZ7i
QmNSu66sGJxiFbdmowhA2FmedJBiyjV6x521DjAKUiu3LH
QmNTZy7TfXvsHczwwV3EYbxRZN5gthgicG9GiroD7C4ZrP
QmNUCfyL67rqBesyCLon9kRUpXtPEfVUzf7aGp1VXqGqW4
QmNV721hfbzxuLdBvLaBpzmKtnPP4duf59SZ3AU1fV5CrK
...
```

### 1.6.8 PubSub En línea

Un cop apreses les comandes més bàsiques de IPFS, entrem en temes més interessants que es poden aplicar amb l'ús de les comandes de IPFS:

Necessitem tenir almenys 3 terminals oberts, 1 per activar el daemon de pubsub, un per a crear un Publicador de tòpics i un per el Subscriptor del tòpic:

Activem el daemon per a pubsub, encara en experiment:

```
1>ipfs daemon --enable-pubsub-experiment
Initializing daemon...
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/169.254.128.121/tcp/4001
Swarm listening on /ip4/169.254.229.236/tcp/4001
Swarm listening on /ip4/192.168.1.41/tcp/4001
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /p2p-circuit/ipfs/QmTrYg8LBa7WwgCp6SqDdH3
xHk2Z4iCKC6zCKJuNyLjuLQ
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/169.254.128.121/tcp/4001
Swarm announcing /ip4/169.254.229.236/tcp/4001
Swarm announcing /ip4/192.168.1.41/tcp/4001
Swarm announcing /ip6/::1/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

Activem primer el Subscriptor per a què observi tots els tòpics del Publicador:

```
2>ipfs pubsub sub tfg
```

I seguidament el Publicador

```
3> ipfs pubsub pub tfg Estudi del sistema distribuit IPFS
```

Si observem en el terminal del Subscriptor, observarem el contingut enviat pel Publicador:

```
>ipfs pubsub sub tfg
Estudi del sistema distribuit IPFS
```

## 1.7 Sistemes relacionats

### 1.7.1 Filecoin

Filecoin [6] és un servei creat per Juan Benet basat en una xarxa d'emmagatzematge d'arxius descentralitzada basat en el mateix concepte que IPFS amb l'objectiu de proporcionar un servei d'emmagatzematge actiu, és a dir per a tenir arxius permanentment disponibles.
Per a poder utilitzar el servei d'emmagatzematge és necessari pagar mitjançant la moneda virtual Filecoin i, per contrapartida, tot usuari que ofereixi emmagatzematge disponible en tot moment al sistema, anomenat miner, o s'ofereixi com a punt d'enviament d'arxius del sistema, obtindrà Filecoins.
La moneda virtual Filecoin pot ser intercanviada per a altres monedes del mercat, com és el Bitcoin o el Ethers. El servei de Filecoin, de la mateixa manera que IPFS, permet tenir un sistema distribuït, escalable i anònim per a emmagatzemar arxius al núvol del sistema. La idea més important de Filecoin respecte la majoria d'empreses que venen emmagatzematge al Cloud o núvol, a part de garantir accesibilitat de les dades i seguretat, les dades emmagatzemades no depenen directament del proveïdor i estan en uns servidors concrets, sinó que està emmagatzemat en les diferents màquines que proporcionen el servei d'emmagatzematge.

#### Característiques

- Tot usuari o empresa amb connexió a internet pot fer ús del servei Filecoin
- El contingut de la xarxa es replica entre els diversos nodes.
- El contingut s'emmagatzema encriptat i els miners no tenen accés al contingut ja que no disposen de la clau per a desencriptar els arxius.

#### IPFS i Filecoin

Un dels problemes que pot tenir IPFS és el Pin dels objectes de la xarxa, que pot comportar la pèrdua de l'emmagatzematge dels arxius quan el Garbage Collector s'activa. Si conbinem IPFS amb Filecoin, obtenim una xarxa distribuïda permanentment activa i sense pèrdua d'arxius ja que amb Filecoin pagues un servei d'emmagatzematge actiu i sempre podràs obtenir els teus fitxers de un dels nodes del sistema distribuït de forma segura.

Els usuaris que emmagatzemen fitxers a IPFS i Filecoin es beneficien dels costos d'emmagatzematge reduïts, d'una diversitat de proveïdors d'emmagatzematge i de serveis de dades molt més ràpids.

## 1.8 Conclusions i desenvolupament futur

Per a concloure i revisar el conjunt de treball, IPFS aporta una solució d'emmagatzematge de fitxers distribuït simple però inovadora alhora, garantint als usuaris seguretat i fiabilitat dins un sistema organitzat amb milers d'usuaris actius.

Per tal de mantenir un fitxer actiu, o "pinnejat"al sistema de IPFS, s'ha descrit una solució econòmica basada en Filecoin, una moneda virtual de pagament per a poder garantir l'accessibilitat als fitxers en qualsevol moment.

### 1.8.1 Idees pera desenvolupar

Amb la solució que aporta IPFS hi ha infinitats d'idees que poden aportar solucions reals, algunes idees es descriuen a continuació:
- Subsistema de fitxers distribuït entre metge i pacient per a obtenir anàlisis o prospectes de forma segura. Per tal de fer-ho possible, cada pacient necessitaria tenir un usuari amb el nodeID públic i el metge sols hauria de pujar l'arxiu al sistema i encriptar-lo amb el nodeID de el pacient per tal que només ell pugués disposar de l'informació malgrat que tot el sistema sería capaç de descarregar-se el fitxer.
- Crear una web que es mantingui activa sense estar centralitzada en un servidor central, com es va fer l'1 d'octubre de 2017, per tal de garantir la llibertat de expressió a la xarxa i que no es pugui atacar directament sinó que sigui necessari fer caure tots els nodes que disposen de la web per a fer-la caure.
- A partir de l'estudi actual, fer un estudi sobre la velocitat del sistema IPFS en diversos entorns, analitzar la seguretat del sistema, entre d'altres.

## 1.9 Budget

El Treball de Fi de Grau en el grau de ENGINYERIA DE TECNOLOGIES I SERVEIS DE TELECOMUNICACIÓ equival a 18 crèdits ECTS, els quals corresponen a un total de 450 hores dedicades (25 hores per ECTS).

### 1.9.1 Cost Recurs Humà

Per a fer la següent evaluació econòmica del projecte, s'està tenint en compte que una empresa paga per un enginyer Junior 1800€ nets mensuals en una jornada laboral de 8h durant 5 dies a la setmana, o 160 hores mensuals que equival a 11,25€/hora.
Amb la hipòtesis indicada, el present treball té un cost de 5.062,50€.
Per a fer el treball s'ha utilitzat programari lliure, Ubuntu com a sistema operatiu i LaTeX per a versionar i descriure el treball.

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

# Revision history and approval record

| Revision | Date | Purpose |
|---|---|---|
| 0 | 17/09/2018 | Document creation |
| 1 | 25/01/2019 | Document revision |
| | | |
| | | |
| | | |

DOCUMENT DISTRIBUTION LIST

| Name | e-mail |
|---|---|
| Jordi Salarich i Martínez | jordi.sala.martinez@alu-etsetb.upc.edu |
| Jose Luis Muñoz i Tapia | jose.luis.muñoz@upc.edu |
| | |
| | |
| | |
| | |

| Written by: | | Reviewed and approved by: | |
|---|---|---|---|
| Name | Jordi Salarich I Martinez | Name | Jose Luis Muñoz i Tapia |
| Position | Project Author | Position | Project Supervisor |

# Índex de figures

1.1 Sistema centralitzat. . . . . 6
1.2 Sistema descentralitzat. . . . . 7
1.3 Sistema distribuït. . . . . 8
1.4 Model simple del funcionament del protocol HTTP . . . . . 9
1.5 DHT - Taula de Hash distribuïda . . . . . 12
1.6 Arbre binari de Kademlia-DHT . . . . . 13
1.7 Figura on es mostra l'element més recent dins un k-Bucket . . . . . 13
1.8 Arxiu amb un volum superior a 256kb subdividit i organitzat en un Merkle DAG . 16
1.9 Fitxer subdividit en diversos blocs . . . . . 17
1.10 Directori amb diversos fitxers . . . . . 18
1.11 Exemple - Estructura de carpetes . . . . . 27

.

# Bibliografia


[1] JUAN BENET ".IPFS - CONTENT ADDRESSED, VERSIONED, P2P FILE SYSTEM", 2014.

[2] O. ESPARZA. "P2P NETWORKS. NETWORK ENGINEERING", UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC), 2015

[3] P. MAYMOUNKOV AND D. MAZIERES. "KADEMLIA: A PEER-TO-PEER INFORMATION SYSTEM BASED ON THE XOR METRIC". IN PEER-TO-PEER SYSTEMS, SPRINGER, 2002.

[4] I. BAUMGART AND S. MIES. "S/KADEMLIA: A PRACTICABLE APPROACH TOWARDS SECURE KEY-BASED ROUTING. IN PARALLEL AND DISTRIBUTED SYSTEMS", 2007

[5] M. J. FREEDMAN, E. FREUDENTHAL, AND D. MAZIERES. "DEMOCRATIZING CONTENT PUBLICATION WITH CORAL", NEW YORK UNIVERSITY, 2004

[6] JUAN BENET. "FILECOIN. A DECENTRALIZED STORAGE NETWORK", 2017

[7] POUWELSE, JOHAN; "THE BITTORRENT P2P FILE-SHARING SYSTEM: MEASUREMENTS AND ANALYSIS", 2005

[8] BRIAN NEIL LEVINE CLAY SHIELDS N. BORIS MARGOLIN ".A SURVEY OF SOLUTIONS TO THE SYBIL ATTACK",2017