Reti e Laboratorio III
Modulo Laboratorio III
AA. 2022-2023

docente: Laura Ricci

laura.ricci@unipi.it

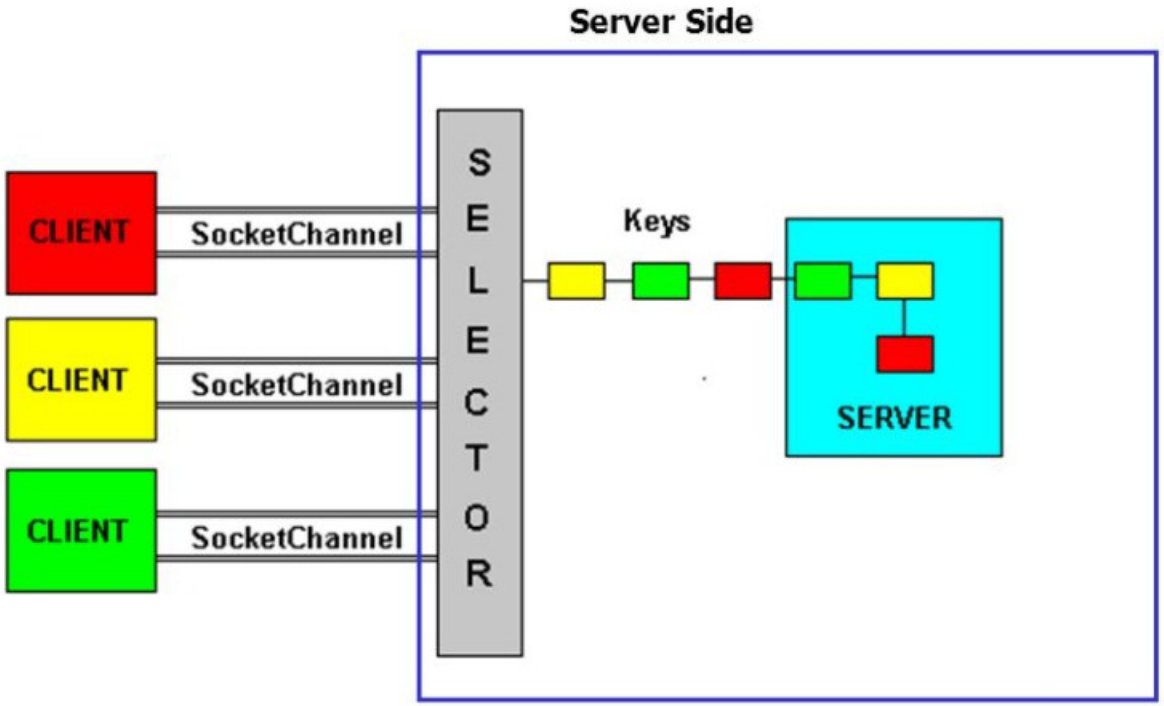
Lezione 11
Esercitazione
01/12/2022

NIO SELECTOR: PATTERN GENERALE

```
Set<SelectionKey> selectedKeys = selector.selectedKeys();
Iterator <SelectionKey> keyIterator = selectedKeys.iterator();
while(keyIterator.hasNext()) {
    SelectionKey key = (SelectionKey) keyIterator.next();
    keyIterator.remove();
    if(key.isAcceptable()) {
        // a connection was accepted by a ServerSocketChannel.
         }
    else if (key.isConnectable()) {
        // a connection was established with a remote server.
        }
        else if (key.isReadable()) {
               // a channel is ready for reading
        }
        else if (key.isWritable()) {
        // a channel is ready for writing }
```

NIO SELECTOR: PATTERN GENERALE

- iterazione sull'insieme di chiavi che individuano i "canali pronti"
- dalla chiave si può ottenere un riferimento al canale su cui si è verificato l'evento
- `keyIterator.remove()` deve essere invocata, poiché il `Selector` non rimuove le chiavi



SELECTION KEY: L'ATTACHMENT

- attachment: riferimento ad un generico `Object`
- utile quando si vuole accedere ad informazioni relative al canale (associato ad una chiave) che riguardano il suo stato pregresso
- necessario perchè le operazioni di lettura o scrittura non bloccanti non possono essere considerate atomiche: nessuna assunzione sul numero di bytes letti
- consente di tenere traccia di quanto è stato fatto in una operazione precedente.
  - l'attachment può essere utilizzato per accumulare i byte restituiti da una sequenza di letture non bloccanti
  - memorizzare il numero di bytes che si devono leggere in totale.

NIO MULTIPLEXED INTEGER GENERATION SERVICE

- sviluppare un servizio di generazione di una sequenza di interi il cui scopo è testare l'affidabilità della rete, mediante generazione di numeri binari
- quando il server è contattato dal client, esso invia al client una sequenza di interi rappresentati su 4 bytes

```
0, 1, 2, ...
```

- il server genera una sequenza infinita di interi
- il client interrompe la comunicazione quando ha ricevuto sufficiente informazioni

NIO MULTIPLEXED INTEGER GENERATION SERVICE

```
import java.nio.*;
import java.nio.channels.*;
import java.net.*;
import java.util.*;
import java.io.IOException;
public class IntGenServer {
  public static int DEFAULT_PORT = 1919;
  public static void main(String[] args) {
        int port;
        try {
             port = Integer.parseInt(args[0]);
             }
        catch (RuntimeException ex) {port = DEFAULT_PORT; }
        System.out.println("Listening for connections on port " + port);
```

NIO MULTIPLEXED INTEGER GENERATION SERVICE

```
ServerSocketChannel serverChannel;
Selector selector;
    try {
        serverChannel = ServerSocketChannel.open();
        ServerSocket ss = serverChannel.socket();
        InetSocketAddress address = new InetSocketAddress(port);
        ss.bind(address);
        serverChannel.configureBlocking(false);
        selector = Selector.open();
        serverChannel.register(selector, SelectionKey.OP_ACCEPT);
       }
    catch (IOException ex) {
            ex.printStackTrace();
            return;
                              }
```

NIO MULTIPLEXED INTEGER GENERATION SERVICE

```
while (true) {
  try {
       selector.select();
      } catch (IOException ex) {
              ex.printStackTrace();
               break;
                                  }
 Set <SelectionKey> readyKeys = selector.selectedKeys();
 Iterator <SelectionKey> iterator = readyKeys.iterator();
```

NIO MULTIPLEXED INTEGER GENERATION SERVICE

```
while (iterator.hasNext()) {
   SelectionKey key = iterator.next();
   iterator.remove();
   // rimuove la chiave dal Selected Set, ma non dal Registered Set
   try {
     if (key.isAcceptable()) {
        ServerSocketChannel server = (ServerSocketChannel) key.channel();
        SocketChannel client = server.accept();
        System.out.println("Accepted connection from " + client);
        client.configureBlocking(false);
        SelectionKey key2 = client.register(selector,SelectionKey.OP_WRITE);
        ByteBuffer output = ByteBuffer.allocate(4);
        output.putInt(0);
        output.flip();
        key2.attach(output); }
```

NIO MULTIPLEXED INTEGER GENERATION SERVICE

```
else if (key.isWritable())
        { SocketChannel client = (SocketChannel) key.channel();
          ByteBuffer output = (ByteBuffer) key.attachment();
          if (! output.hasRemaining())
             {
                 output.rewind();
                 int value = output.getInt();
                 output.clear();
                 output.putInt(value + 1);
                 output.flip();
              }
          client.write(output);}
       } catch (IOException ex) { key.cancel();
                          try { key.channel().close(); }
                          catch (IOException cex) {}      }}}}}
```

NIO INTEGER GENERATION CLIENT

```
import java.nio.*;
import java.nio.channels.*;
import java.net.*;
import java.io.IOException;
public class IntGenClient {
     public static int DEFAULT_PORT = 1919;
     public static void main(String[] args) {
         if (args.length == 0) {
           System.out.println("Usage: java IntgenClient host [port]");
           return;   }
         int port;
         try {
               port = Integer.parseInt(args[1]);
              } catch (RuntimeException ex) {
                       port = DEFAULT_PORT;
                                          }
```

NIO INTEGER GENERATION CLIENT

```
try { SocketAddress address = new InetSocketAddress(args[0], port);
      SocketChannel client  = SocketChannel.open(address);
      ByteBuffer    buffer  = ByteBuffer.allocate(4);
      IntBuffer     view    = buffer.asIntBuffer();
      for (int expected = 0;  ; expected++) {
        client.read(buffer);
        int actual = view.get();
        buffer.clear();
        view.rewind();
        if (actual != expected) {
          System.err.println("Expected " + expected + "; was " + actual);
          break;
        }
        System.out.println(actual);
      }
    } catch(IOException ex) { ex.printStackTrace(); } }  }
```

SSDP SIMPLE SERVICE DISCOVERY PROTOCOL (SSDP)

- su una rete locale attivi molti protocolli che usano il multicast
- SSDP attivo sulla porta 1900 associato all'indirizzo di multicast 239.255.255.250
- un protocollo di discovery usato per scoprire quali servizi sono disponibili in una rete
- ogni servizio è definito mediante specifiche UPnP
- il server SSDP invia pacchetti di advertisement sul gruppo di multicast
- formato di un pacchetto di advertisement

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age=120
ST: urn:schemas-upnp-org:device:WANDevice:1
USN: uuid:fc4ec57e-b051-11db-88f8-
0060085db3f6::urn:schemas-upnp-org:device:WANDevice:1
EXT:
SERVER: Net-OS 5.xx UPnP/1.0
LOCATION: http://192.168.0.1:2048/etc/linuxigd/gatedesc.xml
```

SSDP SIMPLE SERVICE DISCOVERY PROTOCOL (SSDP)

Messaggio di advertisement

- USN, Unique Service Name
  - UUID (Universally Unique Identifier) che identifica una device o un servizio
- LOCATION
  - punta ad una URL
  - a quella URL l'utente può trovare una descrizione XML delle capability del servizio

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age=120
ST: urn:schemas-upnp-org:device:WANDevice:1
USN: uuid:fc4ec57e-b051-11db-88f8-
0060085db3f6::urn:schemas-upnp-org:device:WANDevice:1
EXT:
SERVER: Net-OS 5.xx UPnP/1.0
LOCATION: http://192.168.0.1:2048/etc/linuxigd/gatedesc.xml
```

SNIFFING SSDP IN JAVA

```
import java.io.*; import java.net.*;
public class MulticastSniffer {
 public static void main(String[] args) {
    InetAddress group = null;
    int port = 0;
    // read the address from the command line
    try {
      group = InetAddress.getByName(args[0]);
      port = Integer.parseInt(args[1]);
    } catch (ArrayIndexOutOfBoundsException | NumberFormatException |
                                                    UnknownHostException ex)
        {
           System.err.println( "Usage: java MulticastSniffer multicast_address
                                                    port");
           System.exit(1); }
    MulticastSocket ms = null;
```

SNIFFING SSDP IN JAVA

```
try {ms = new MulticastSocket(port);
     ms.joinGroup(group);
     byte[] buffer = new byte[8192];
     while (true) {
        DatagramPacket dp = new DatagramPacket(buffer, buffer.length);
        ms.receive(dp);
        String s = new String(dp.getData(), "8859_1");
        System.out.println(s); }
 } catch (IOException ex) { System.err.println(ex);
 } finally {
   if (ms != null) {
     try {
       ms.leaveGroup(group);
       ms.close(); } catch (IOException ex) { } } } } }
```

SNIFFING SSDP IN JAVA: OUTPUT GENERATO

```
$ Java MulticastSniffer 239.255.255.250 1900

NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age=1801
NTS: ssdp:alive
LOCATION: http://192.168.1.1:49152/gKgq6lu34h/wps_device.xml
SERVER: Unspecified, UPnP/1.0, Unspecified
NT: urn:schemas-wifialliance-org:service:WFAWLANConfig:1
USN: uuid:03fef68b-319f-5ea7-80a8-2aea5bb7e216::urn:schemas-
wifialliance-org:service:WFAWLANConfig:1
…….
```

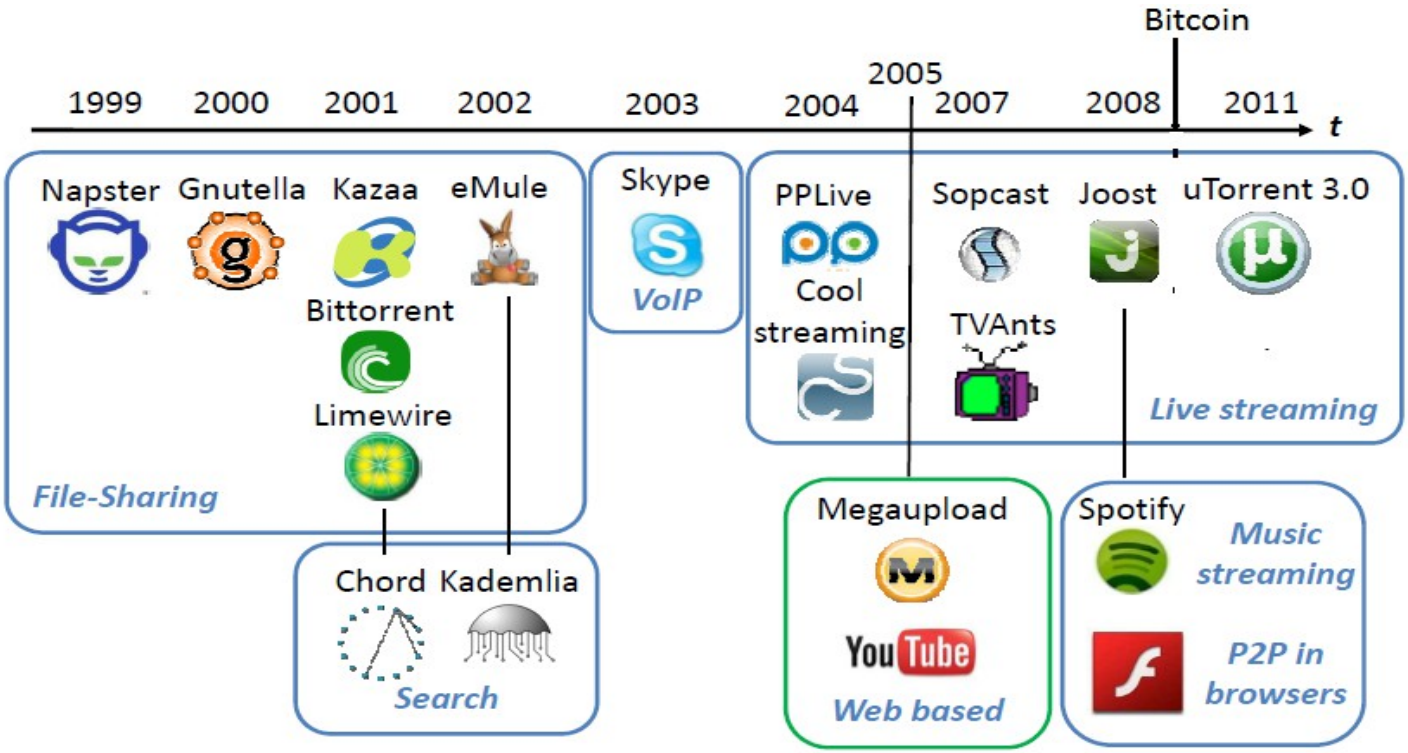
ASSIGNMENT

Definire un Server `TimeServer`, che

- invia su un gruppo di multicast dategroup, ad intervalli regolari, la data e l'ora.
- attende tra un invio ed il successivo un intervallo di tempo simulata mediante il metodo `sleep()`.
- l'indirizzo `IP` di dategroup viene introdotto da linea di comando.
- definire quindi un client `TimeClient` che si unisce a dategroup e riceve, per dieci volte consecutive, data ed ora, le visualizza, quindi termina.

IL MODELLO PEER TO PEER

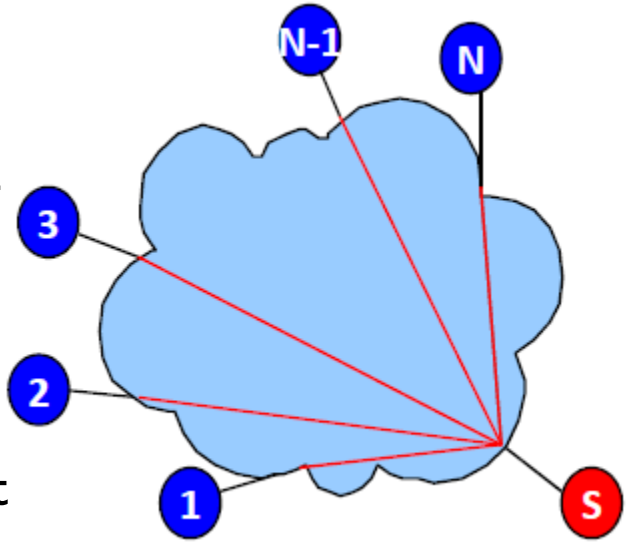
- in questo corso abbiamo analizzato il modello client server, come base per costruire applicazioni di rete
- molte applicazioni sono sviluppate secondo un modello alternativo: il modello peer to peer: nato nel 2000 per applicazioni di file-sharing



IL PARADIGMA CLIENT SERVER

- in esecuzione sugli end host
- comportamento on/off
- utilizza servizi
- inoltra richieste
- nessuna interazione tra client
- deve conoscere un riferimento al servizio

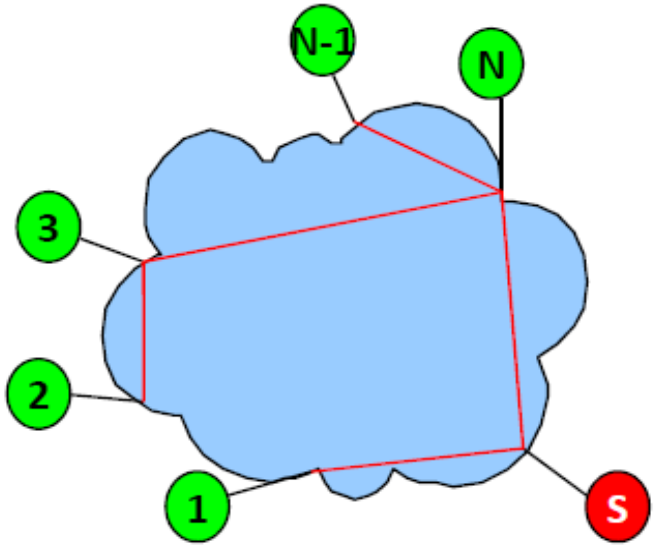
- in esecuzione su un host dedicato
- "always on"
- fornisce servizi
- riceve richieste dai client
- soddisfa le richieste dell'utente
- deve avere un IP fisso (o DNS name)



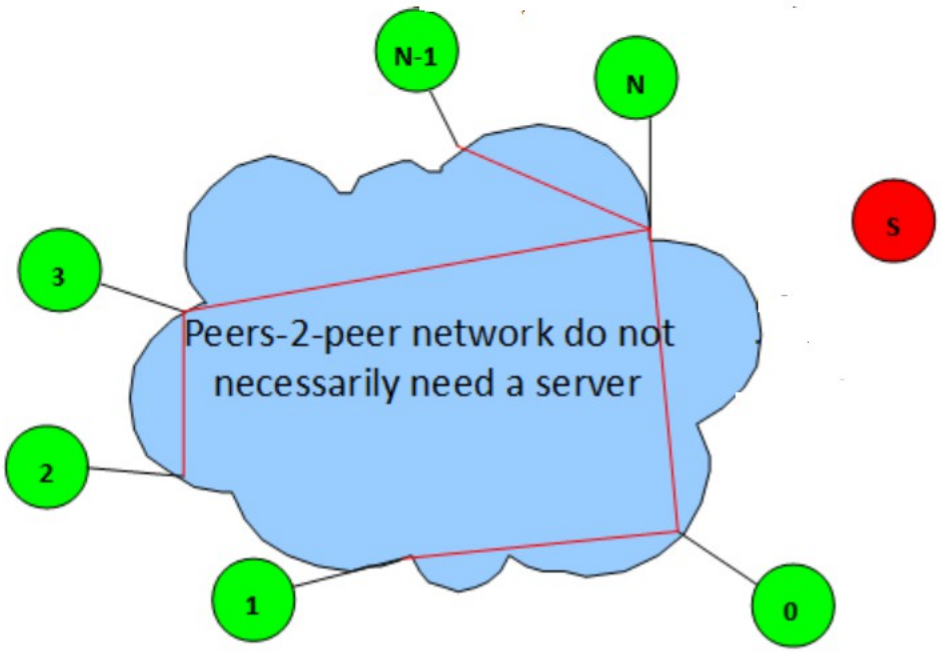
IL PARADIGMA P2P

Peer

- in esecuzione sugli end-hosts
- produttori e consumatori di servizi
- comportamento on/off
- alto livello di dinamicità (churn)
- necessaria una fase di bootstrap
- necessari meccanismi per la scoperta di peer
- comunicano tra di loro
- necessari protocolli a livello applicazione per
  - evitare il fenomeno dei `free riders`
  - incentivare partecipazione e collaborazione



IL PARADIGMA P2P



notare che:

- un server è sempre presente, ma serve solo nella fase di bootstrap
- i server non sono necessari per la condivisione delle risorse

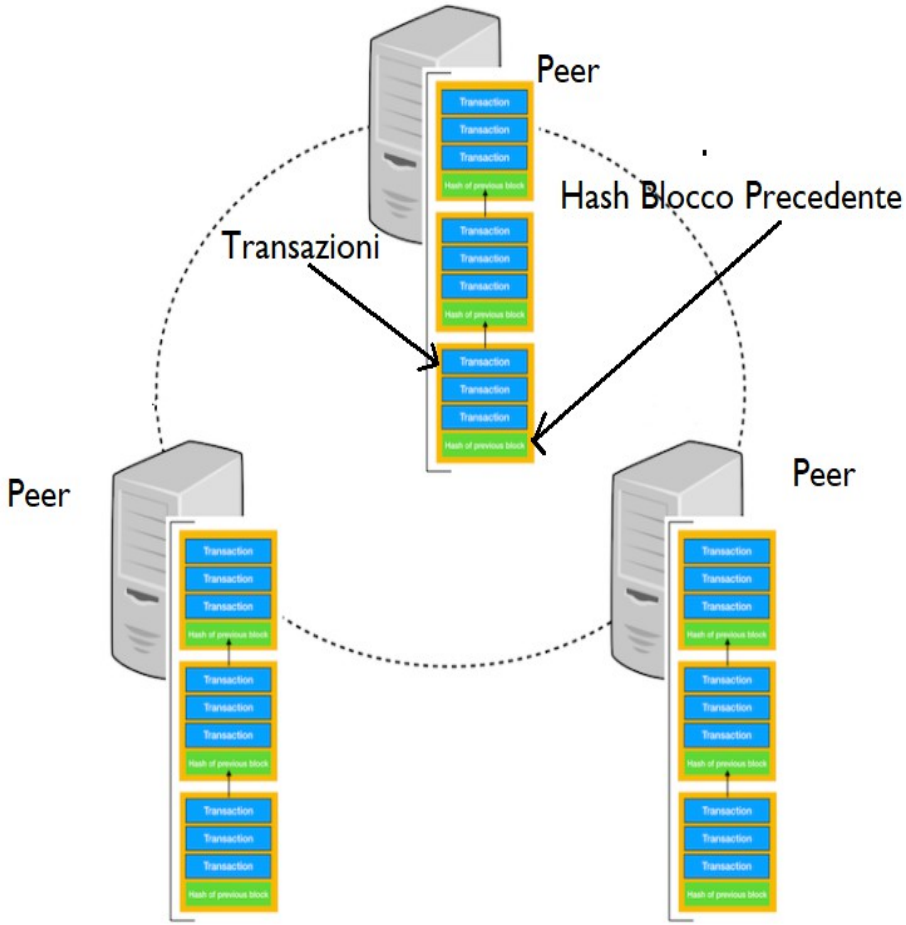
P2P: LE APPLICAZIONI

- P2P file sharing
  - file sharing: light weight/ best effort
  - la persistenza e la sicurezza non sono l'obiettivo principale
  - l'anonimato è importante
  - applicazione
    - Napster
    - Gnutella, KaZaa
    - eMule
    - BitTorrent
- P2P Media Streaming
  - Skype (first versions)
- Cyptocurrencies and blockchains
- Distributed file System: Internet Planetary File System (IPFS)

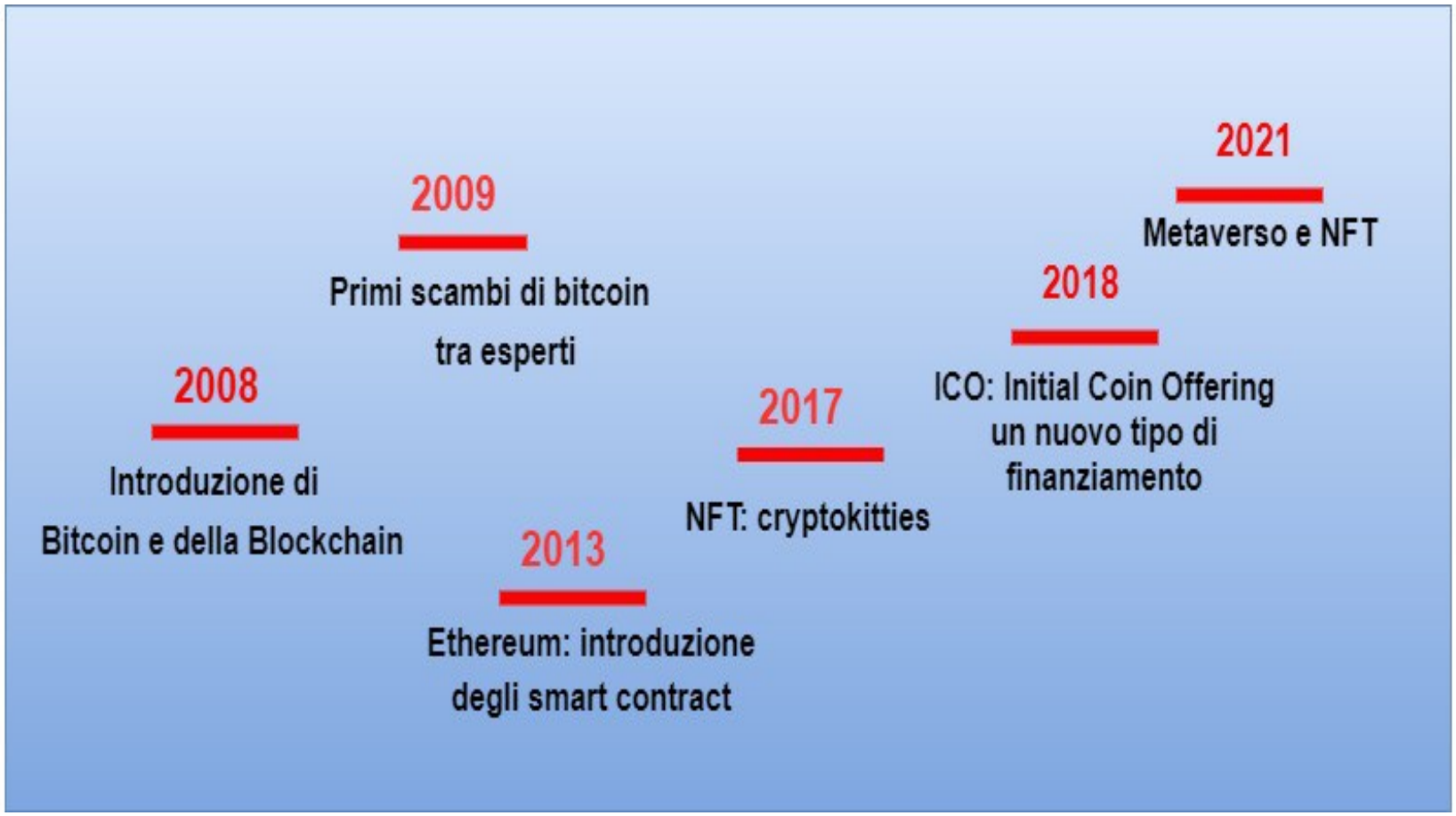
BLOCKCHAIN IN BREVE

- database distribuito e replicato sui nodi di un sistema peer to peer (P2P)
  - un insieme di blocchi collegati mediante puntatori hash
  - in ogni blocco è presente l'hash blocco precedente
  - ogni peer possiede una copia consistente dell'intero database

- operazioni
  - append only: accodare progressivamente registrazioni organizzate in blocchi
  - leggere il contenuto di una qualsiasi registrazione



LE BLOCKCHAIN: EVOLUZIONE



L'ECOSISTEMA DI BITCOIN

- non solo blockchain, ma diversi attori coinvolti
  - `Exchangers`, `Wallets`, `NFT Marketplaces`
- una vera e propria economia basata su cryptocurrencies


50+ BLOCKCHAIN
REAL WORLD USES CASES
GOVERNMENT
Essentia develops world's first blockchain solution to manage international logistics hub together with Traffic Labs and the Finnish Government
essentia.one
IDENTIFICATION
Voter registration is being facilitated via a blockchain project in Switzerland spearheaded by Uport.
uport
MOBILE PAYMENTS
The blockchain ledger that Ripple uses has been latched onto by a group of Japanese banks, who will be using it for quick mobile payments.
ripple
INSURANCE
A smart contract-based blockchain is being used by Insurer American International Group Inc as a means of saving costs and increasing transparency.
AIG
ENDANGERED SPECIES PROTECTION
The protection of endangered species is being facilitated via a blockchain project that records the activities of these rare animals.
CARBON OFFSETS
IBM is using the Hyperledger Fabric blockchain in China to monitor carbon offset trading.
IBM
HYPERLEDGER
ENTERPRISE
Ethereum's blockchain can be accessed as a cloud-based service courtesy of Microsoft Azure.
Microsoft Azure
BORDER CONTROL
Essentia has devised a border control system that would use blockchain to store passenger data in the Netherlands.
essentia.one
SUPPLY CHAINS
IBM and Walmart have partnered in China to create a blockchain project that will monitor food safety.
IBM
Walmart
HEALTHCARE
A number of healthcare systems that store data on the blockchain have been pioneered including MedRec.
MEDREC
SHIPPING
Shipping is a natural fit for blockchain, and Maersk have been trialling a blockchainbased project within the maritime logistics industry.
MÆRSK
REAL ESTATE
Blockchain is now being used to complete real estate deals, the first of which was conducted in Kiev by Propy.
PROPY
ENERGY
Essentia is developing a test project that will help energy suppliers track the distribution of their resources in real time, whilst maintaining data confidentiality.
essentia.one
LAND REGISTRY
Land registry titles are now being stored on the blockchain in Georgia in a project developed by the National Agency of Public Registry.
PUBLIC REGISTRY
COMPUTATION
Digital Currency Group are helping Amazon Web Services examine ways in which the distributed ledger technology can help improve database security.
DIGITAL CURRENCY GROUP
ADVERTISING
New York Interactive Advertising Exchange has been experimen- ting with blockchain as a means of providing an ads marketplace for publishers.
NYIAX
BORDER CONTROL
Essentia is developing a blockchain project for border control that will allow customs agents to record passenger data from an array of inputs and safely store it.
essentia.one
JOURNALISM
Decentralized journalism, as enabled by blockchain technology, has the potential to prevent censorship and increase transparency, as Civil has shown.
CIVIL
WASTE MANAGEMENT
Waltonchain is using RFID technology to store waste management data on the blockchain in China.
ENERGY
Food importation is another industry where blockchain is proving its worth, with Louis Dreyfus Co trialling a soybean importation operation using this technology.
LDC.
DIAMONDS
The De Beers Group is using blockchain to track the importation and sale of diamonds.
DE BEERS
FINE ART
By storing certificates of authenticity on the blockchain, it's possible to dramatically reduce art forgeries, as one blockchain project is proving.
NATIONAL SECURITY
For the past two years, the US Department of Homeland Security has been using blockchain to record and safely store data captured from its security cameras.
TOURISM
In a bid to boost its tourism economy, Hawaii is examining ways in which blockchain-based cryptocurrencies can be adopted throughout the US state.
TAXATION
In China, a tax-based initiative is using blockchain to store tax records and electronic invoices led by Miaocai Network.
ENERGY
Chile's National Energy Commission has started using blockchain technology as a way of certifying data pertaining to the country's energy usage as it seeks to update its electrical infrastructure.
CNE
COMISIÓN NACIONAL DE ENERGÍA
RAILWAYS
Russian rail operator Novotrans is storing inventory data on a blockchain pertaining to repair requests and rolling stock
НОВОТРАНС
ENTERPRISE
Google is building its own blockchain which will be integrated into its cloud-based services, enabling businesses to store data on it, and to request their own white label version developed by Alphabet Inc
Google
Alphabet
MUSIC
Arbit is a blockchain-based project led by former Guns N Roses drummer Matt Sorum seeking a fairer way to reward musicians for their creative efforts.
arbit
FISHING
Blockchain technology has been used to provide a transparent record of where fish was caught, as a means of ensuring it was legally landed.

P2P FILE SYSTEM: IPFS

Your file, and all of the **blocks within it**, is given a **unique fingerprint** called a **cryptographic hash**.

IPFS **removes duplications** across the network.

Each **network node** stores only content it is interested in, plus some indexing information that helps figure out which node is storing what.

When you **look up a file** to view or download, you're asking the network to find the nodes that are storing the content behind that file's hash.

You don't need to remember the hash, though — every file can be found by **human-readable names** using a decentralized naming system called **IPNS**.

P2P FILE SYSTEM: IPFS

- IPFS: integrazione file system distribuito con blockchain
- dati di grande dimensione memorizzati su IPFS, hash dei dati su blockchain
- implementato in LibP2P
  - una libreria per lo sviluppo di applicazioni P2P
  - nuove tecnologie: Multiprotocol Project
  - Distributed Hash Table: Kademlia

CORSI INSEGNATI SUL TEMA BLOCKCHAIN

- *P2P & Blockchain,* `corso fondamentale, Laurea Magistrale in Informatica, Curriculum ICT`
- *Blockchain e AI,* `corso complementare, Laurea Magistrale Diritto per le nuove tecnologie`
- *laboratorio web scaping*: `corso complementare Laurea Triennale in Informatica`

  focus del corso è sull' analisi di dati provenienti dall'ecosistema delle blockchain
  - parte teorica:
    - richiami di elementi di statistica di base
    - graph models: random graph, power law, small worlds
    - proprietà caratteristiche di un grafo (degree, centrality, clustering coefficient,..)
    - struttura delle transazioni di diversi tipi di blockchain, (Bitcoin, Ethereum,...), transaction graphs
  - laboratorio:
    - uso di API per il reperimento di dati
    - breve introduzione a Python
    - Pandas: Numpy e matplotlib
    - librerie per l'analisi di grafi (NetworkX, Networkkit,...)
    - applicazione degli strumenti a casi s'uso legati all'ecosistame delle blockchain (transazioni, Exchange services, mixers, miners,..)

TESI DI LAUREA TRIENNALI E MAGISTRALI

- applicazione di tecniche crittografiche in collaborazione con la Prof. Bernasconi
  - zero-knowledge
  - authenticated data structures
- analisi di transazioni
  - bot discovery
  - attacchi
  - NFT markets e NFT communities
- layer-2 technologies
  - lightning network: analisi e algoritmi di routing
  - cross-chain solutions: Polygon, ChainBridge
  - oracoli: ChainLink
- applicazioni della blockchain
  - identità digitale: Self Sovereign Identity (SSI)
  - meccanismi di access control mediante smart contract