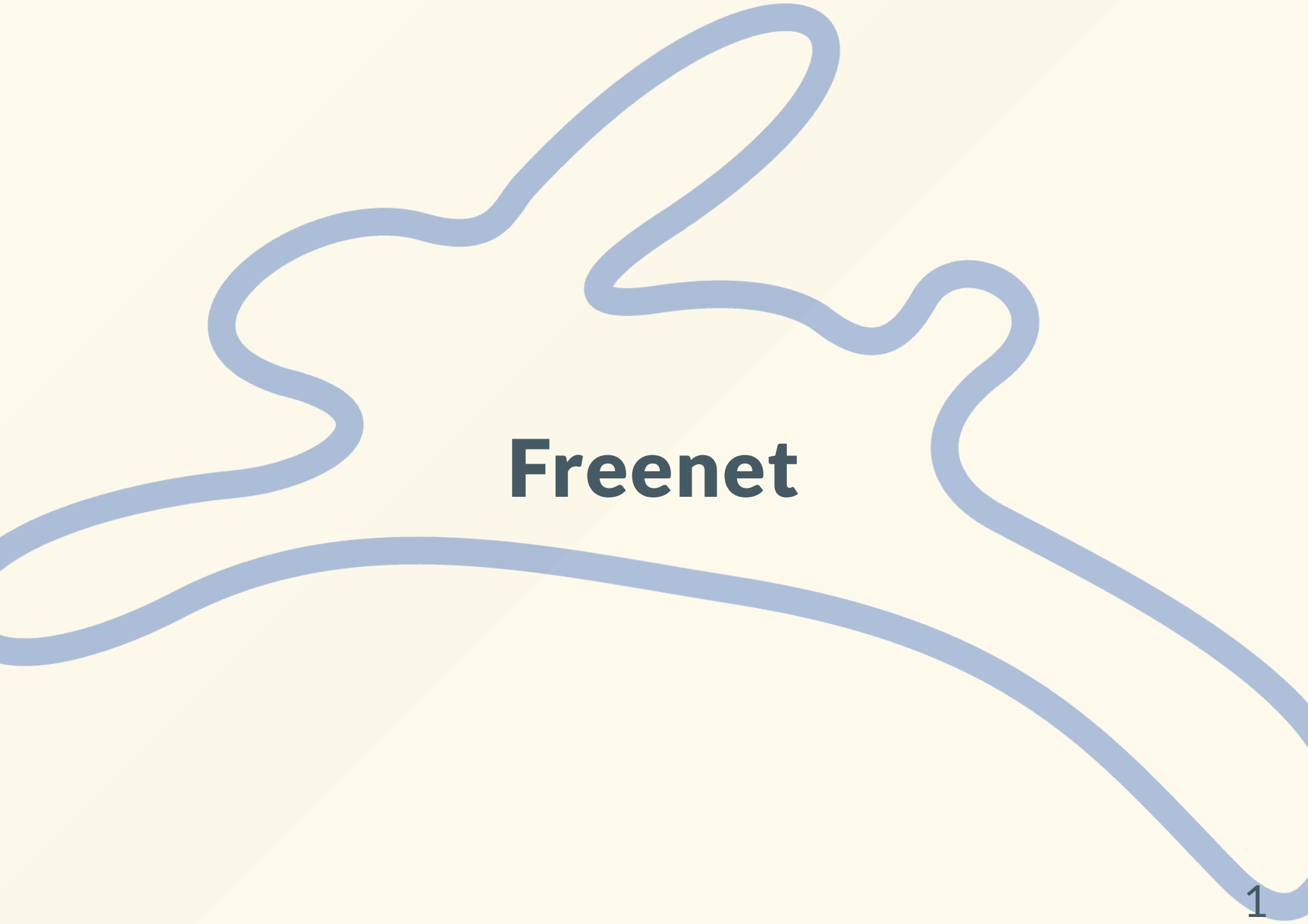


Freenet

Cos'è Freenet?

Freenet è una **rete peer-to-peer decentralizzata**, progettata per:

- Resistere alla censura
- Proteggere l'**anonimato**
- Rispettare la libertà di espressione

Cenni Storici

- Fondazione: **paper** di Ian Clarke, 2001

Il paper contiene le basi del protocollo Freenet, tuttora in sviluppo continuo.

Dal 2008 [**Freenet 0.7**] è divisa in **Opennet** e **Darknet**, supporta l'implementazione **open source** di JDK [**OpenJDK**] e supporta sia su TCP che su UDP a livello di trasporto.

Design Goals

- Anonimato per **produttori e consumatori** di informazioni
- **Resistenza** a tentativi di terze parti di **negare accesso** ai contenuti
- Proteggere l' **anonimità dei dati salvati nei nodi**
- Implementare **archiviazione e routing** in maniera **dinamica** ed **efficiente**
- **Decentralizzare** tutte le funzioni della rete

Architettura

Freenet = Rete di Nodi

Freenet è un network P2P **adattivo**, costituito da nodi che effettuano richieste ad altri nodi per salvare e ottenere **data files**.

- Tutti i nodi sono **uguali** e comunicano **solo con i nodi prossimi**
- Ogni nodo mantiene uno **spazio di archiviazione locale** che **condivide** con gli altri nodi
- I data files sono indicizzati da **chiavi indipendenti dalla localizzazione** *[location-independent keys]*

Protocollo

Il protocollo Freenet:

- È **packet oriented**
- Utilizza **self-contained messages**
- È flessibile sulla **scelta del meccanismo di trasporto** *[TCP, UDP]*
- Utilizza **key-based routing**
- Utilizza uno **spazio di archiviazione decentralizzato**
- Identifica uno **Small World Network**

Indirizzi

Gli indirizzi dei nodi sono composti da:

- Metodo di trasporto
- Identificatore di trasporto specifico *[IP + porta]*

Anonimato e modifica di indirizzi si ottengono con le **Address Resolution Keys** (o **Updatable Subspace Keys**), contenenti l'**indirizzo corrente** del nodo.
Le ARK permettono di superare alcuni firewalls, NAT e blocchi su IP.

USK@<...key...>/path/to/webpage

Richieste

- Richiesta = **payload di un self-contained message**
- Richiesta = **Ricerca di una chiave** corrispondente al data file da ottenere tra i nodi della rete. *[Key-Based Routing]*

La procedura di richiesta avviene come una **catena di proxy requests** divisa in step.

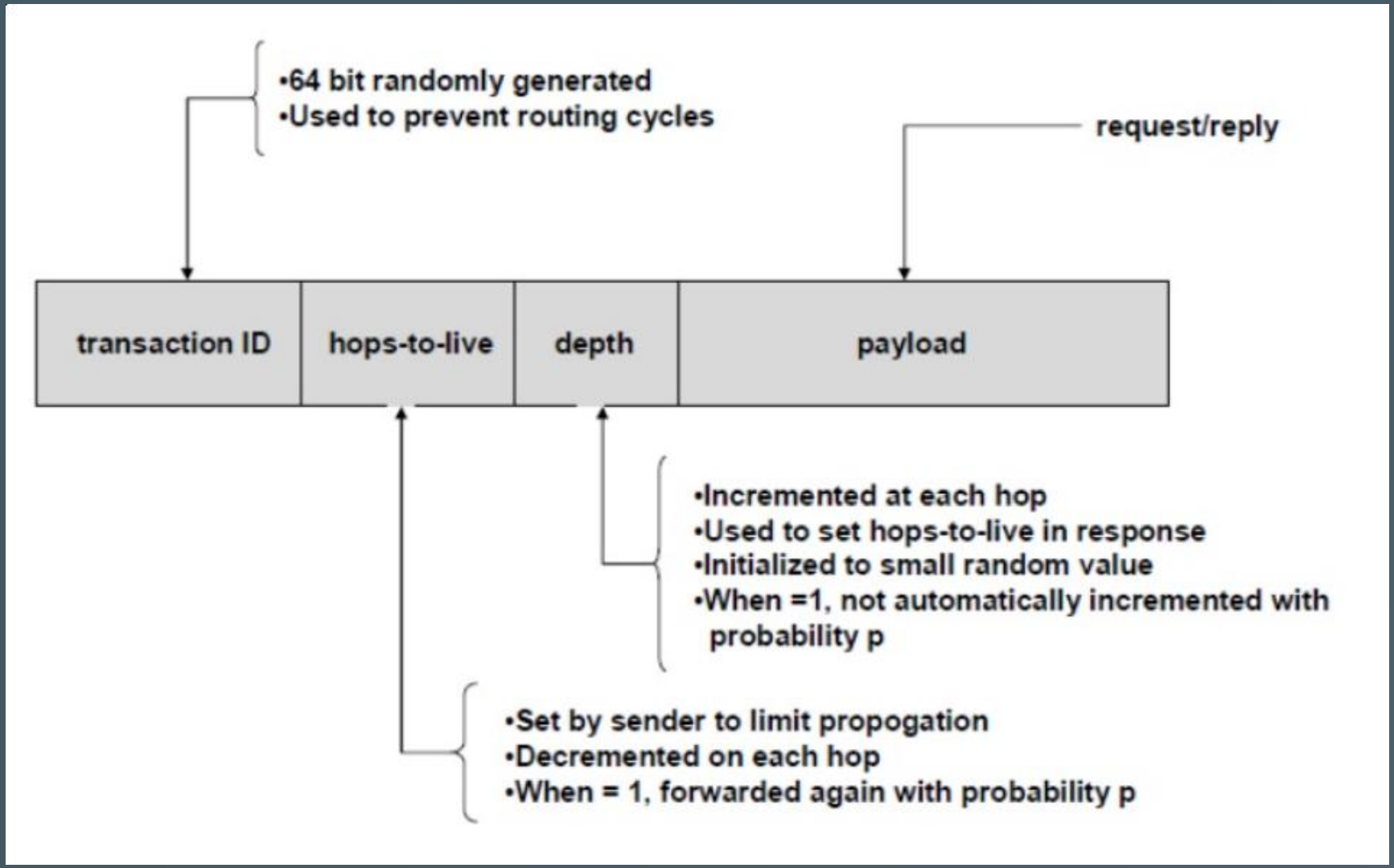
Ogni nodo **decide dove indirizzare la richiesta** durante lo step successivo.

Struttura di un messaggio

- **ID** pseudo-unico e **casuale**, che permette ai nodi di evitare i loop di richieste.
- Numero di **Hops-To-Live** *[HTL]*, simili al Time-To-Live di IP ma basati sul numero di hop. Servono ad **evitare catene di richieste infinite**.
- **Profondità**, incrementata ad ogni hop.
- **Payload** della richiesta / risposta.

Il processo di passaggio delle richieste continua finchè la richiesta non è soddisfatta o gli HTL si esauriscono.

Struttura di un messaggio




Key-Based
Routing

Chiavi e Ricerche

In Freenet, i files sono identificati da **chiavi binarie**:

- **Keyword-Signed-Key** *[KSK]*
- **Signed-Subspace-Key** *[SSK]*
- **Content-Hash-Key** *[CHK]*

Le chiavi sono ottenute utilizzando funzioni di hash tra cui *DSA* e *SHA*.

Keyword Signed Key

La chiave più basica utilizzata da Freenet, derivata da **una stringa di testo scelta dall'utente** al momento del **file upload** in rete.

La stringa è usata come input per ottenere una **coppia di chiavi asimmetriche**, di cui:

- La **pubblica** viene passata alla funzione di hash, ottenendo la **file key**
- La **privata** è utilizzata per firmare il file, provvedendo un **minimo controllo di integrità**.

KSK

Analisi

- Un sistema simile è soggetto a **dictionary attacks** contro la chiave privata, dato che **ogni nodo può ottenere una lista di stringhe descrittive**.

Per questo il file è anche cifrato utilizzando la stringa descrittiva come chiave.

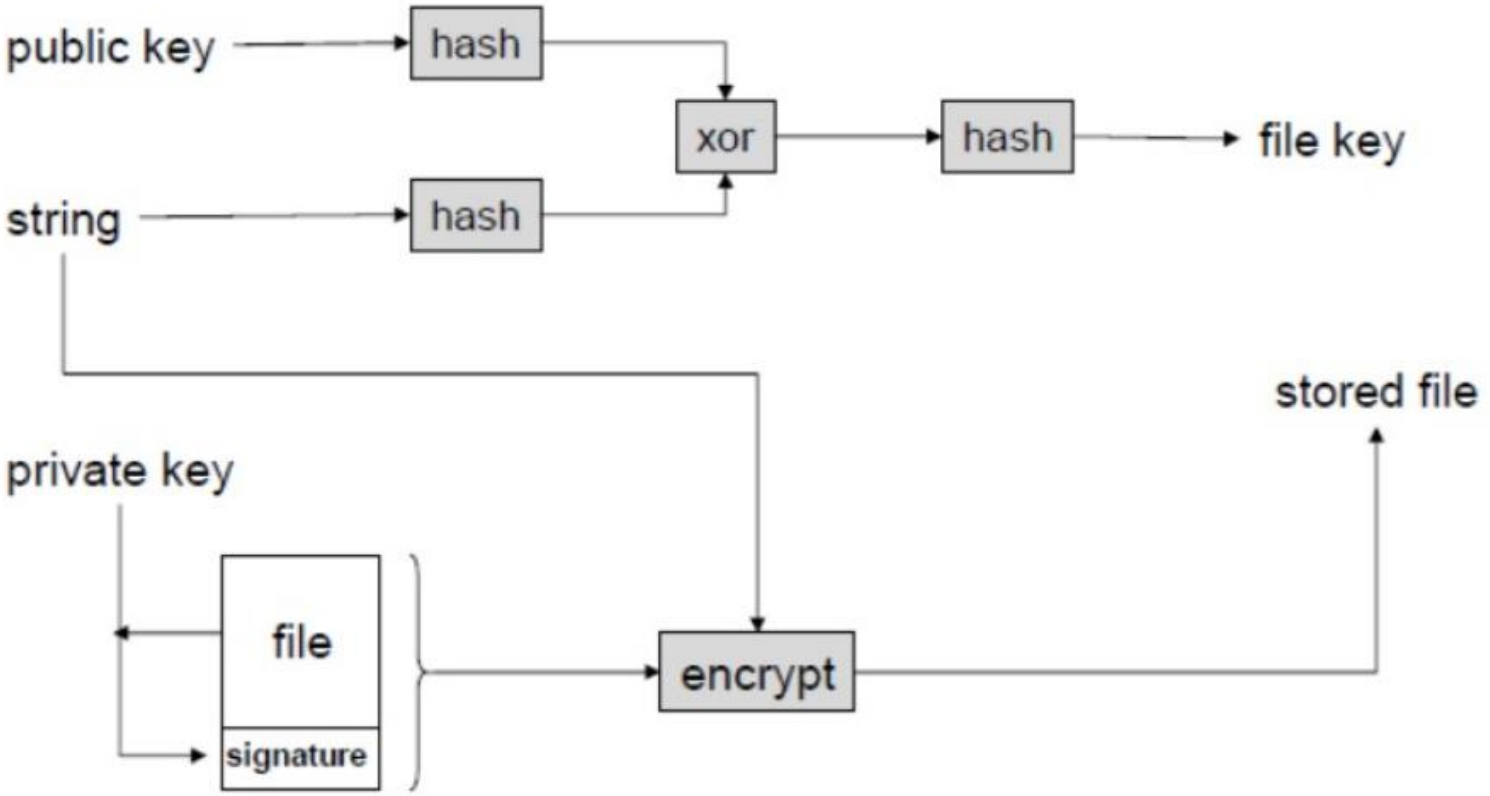
- Le chiavi KSK **non sono uniche**, per questo motivo **sono state abbandonate** in funzione delle SSK.

Signed Subspace Key

Evoluzione di SSK.

- Abilita i **personal namespaces**, identificati da una **coppia di chiavi asimmetriche**
- Stringa descrittiva e chiave pubblica del namespace sono passate alla funzione di hash **indipendentemente**
- Sui due hash ottenuti: **bit-by-bit XOR** -> **file key**
- La chiave privata è utilizzata per firmare il file

Signed-subspace key



SSK

Analisi

- Migliora KSK in quanto **la file key è generata da una coppia di chiavi casuali**

Per ottenere un file, un nodo deve conoscere:

- Stringa descrittiva
- Public Subspace Key

In aggiunta, ogni utente ha un **namespace** unico e gestibile.

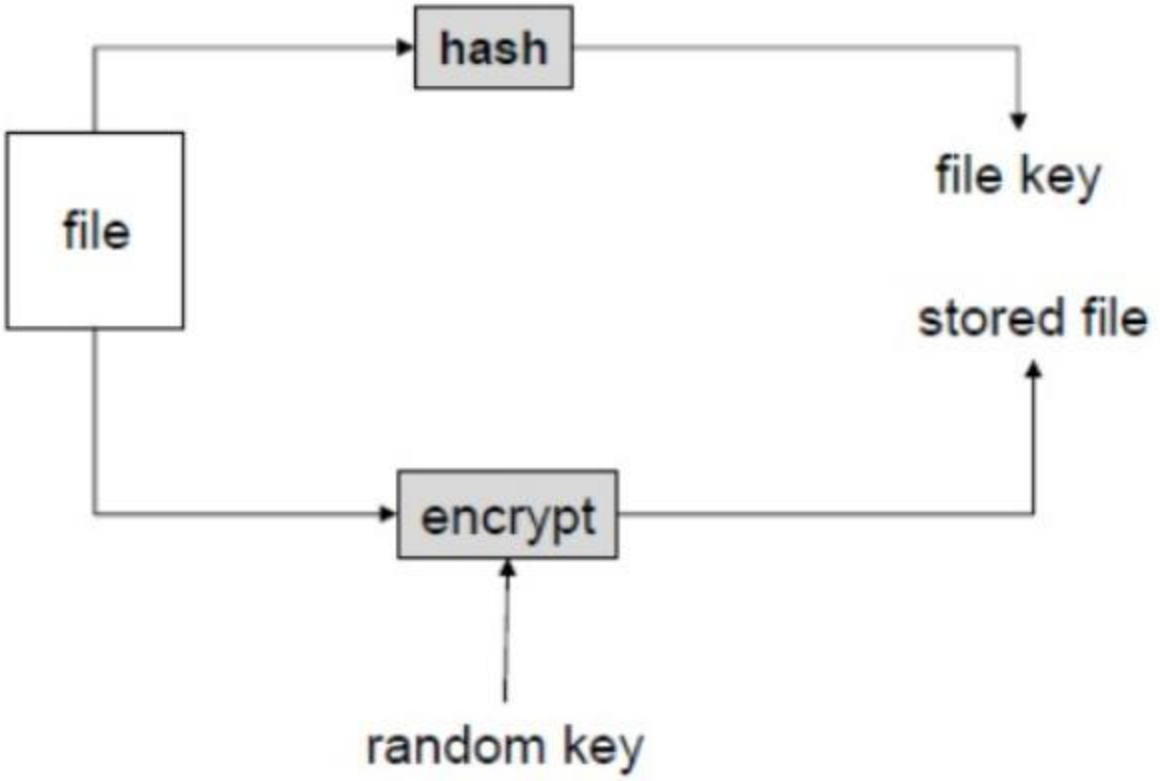
Content Hask Key

Hash diretto del file, ottenendo una **file key pseudo-unica**.

I file sono successivamente cifrati con una ***encryption key*** generata casualmente.

Per ottenere il file sono necessarie sia la file key che l'*encryption key*, che **non è mai salvata col file** ma solo pubblicata con la sua file key.

Content-hash key



Updatable Subspace Key

Le chiavi CHK sono utilizzate **in parallelo** con le SSK, per implementare l'**indirezione**:

- Inserire il file sotto la sua CHK
- Inserire la CHK in un **file indiretto** contenuto in una SSK
- La **file key** è ottenuta in due step.

Questo permette l'**update** dei files caricati, mantenendo la stessa SSK.

CHK + SSK = USK

Decentralized Data Storage

Gestione dei dati salvati

Ogni nodo della rete condivide spazio locale.

- Gestito come una **cache LRU**
  [Least Recently Used cache]
- Contiene **dati** e **routing table** del nodo
- L'utente **non ha accesso diretto** ai file contenuti nel suo spazio locale *[data encryption]*
- La quantità di spazio condiviso è configurabile dall'utente

LRU Cache

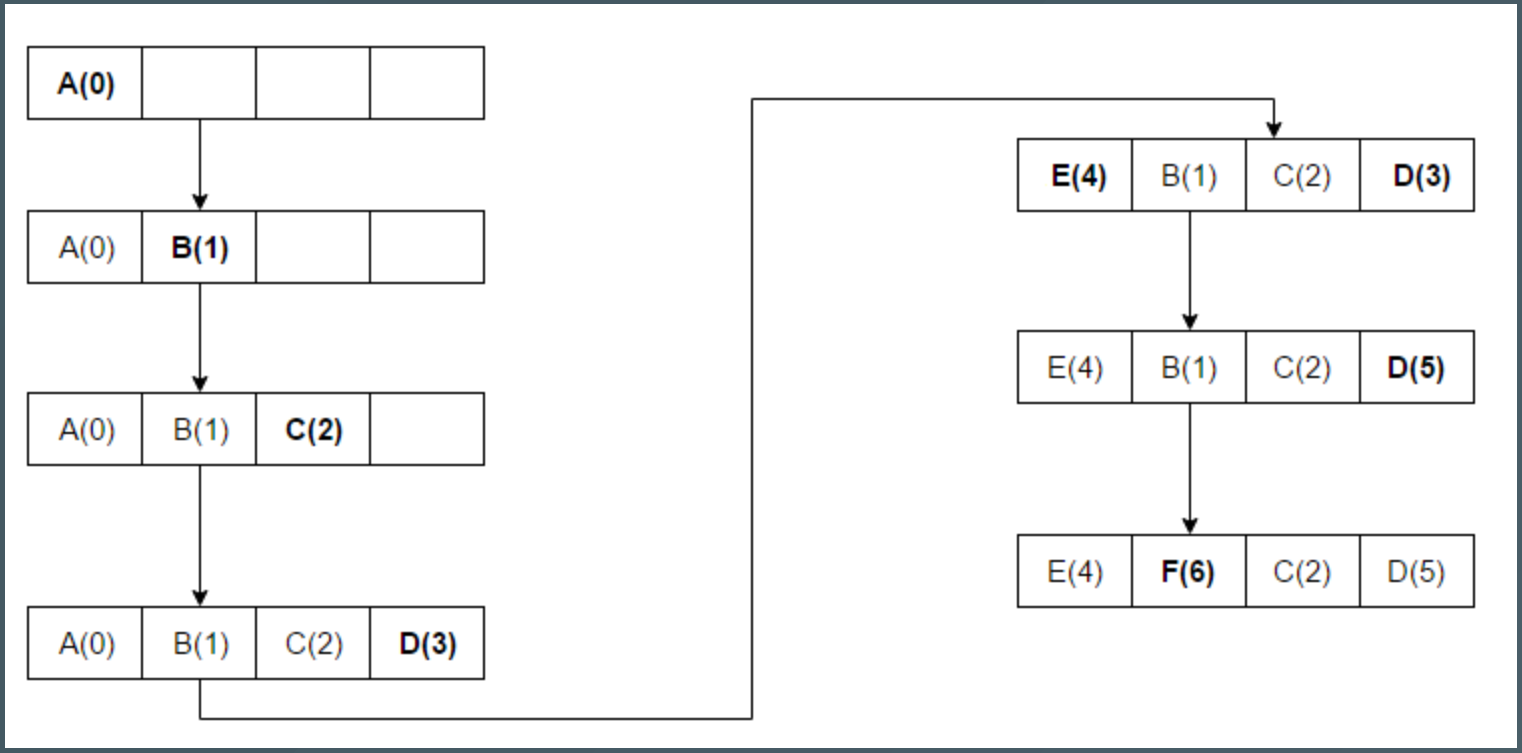
La **Least Recently Used** cache ordina i dati contenuti in base all'**ordine temporale decrescente** di ricevimento delle **richieste**. *[insert,retrieve,update]*

Questo significa che:

- I files **più richiesti** sono in cima alla coda.
- Se la capacità non è sufficiente per contenere un nuovo file, i file usati **meno recentemente** sono cancellati.

LRU Cache - Esempio

La sequenza di accesso è *A,B,C,D,E,F*.
Il **rank** *[priorità]* dei files è il numero in parentesi.



LRU Cache: Note

- La cancellazione dei files utilizzati meno recentemente potrebbe rendere alcuni dati **non disponibili**

Per questo **le entry nella routing table** corrispondenti ai dati cancellati rimangono anche dopo la cancellazione.

- Il meccanismo di ***scadenza*** dei dati causa l'**aggiornamento** dei files vecchi, rimpiazzandoli con le nuove versioni in automatico.

Data Encryption

È necessario che gli operatori dei nodi **non possano accedere** ai dati contenuti nello spazio locale.

Questo è garantito dalla **cifratura** dei files.

- I files sono divisi in **blocchi** distribuiti a diversi nodi
- È possibile conoscere la **file key** ma non la **encryption key** a priori

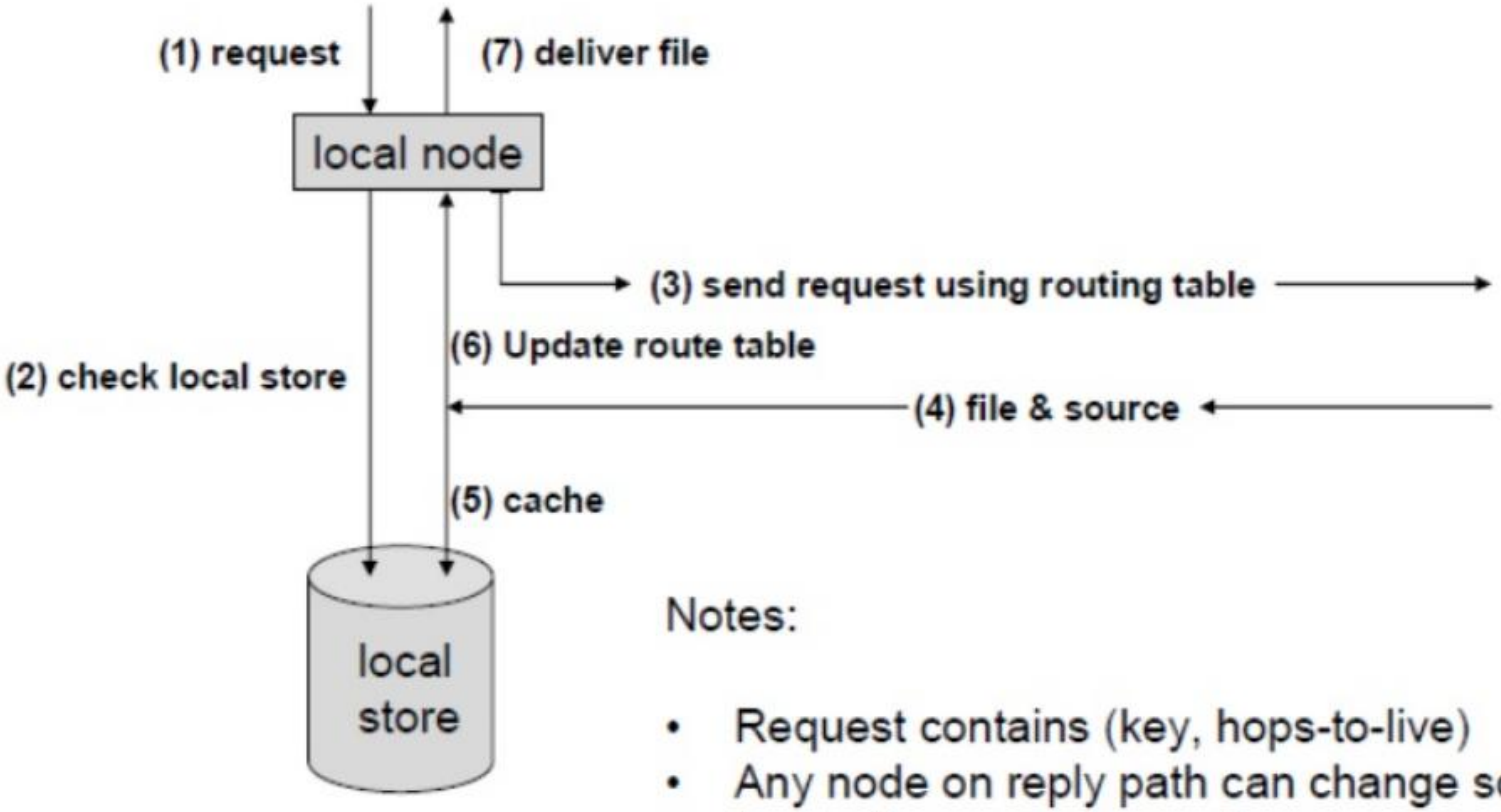
Questo consente la **plausible deniability** per gli operatori dei nodi.

Ottenere un File

Utente manda la **file key** al proprio nodo. Ogni nodo, ricevuta una richiesta, controlla se la chiave è presente nel suo spazio locale.

- **No**: *forward* della richiesta al nodo contenente **la chiave più vicina a quella cercata** della sua forwarding table
- **Sì**: **success**, file mandato ai nodi upstream, ognuno dei quali fa **caching** di esso.
- **HTL expired**: **failure** propagato verso l'upstream, la ricerca si ferma

Ottenere un file



Notes:

- Request contains (key, hops-to-live)
- Any node on reply path can change source to be itself or any other node
- File cached at all nodes along return path
  1. Improved subsequent access
  2. Redundancy improve fault tolerance

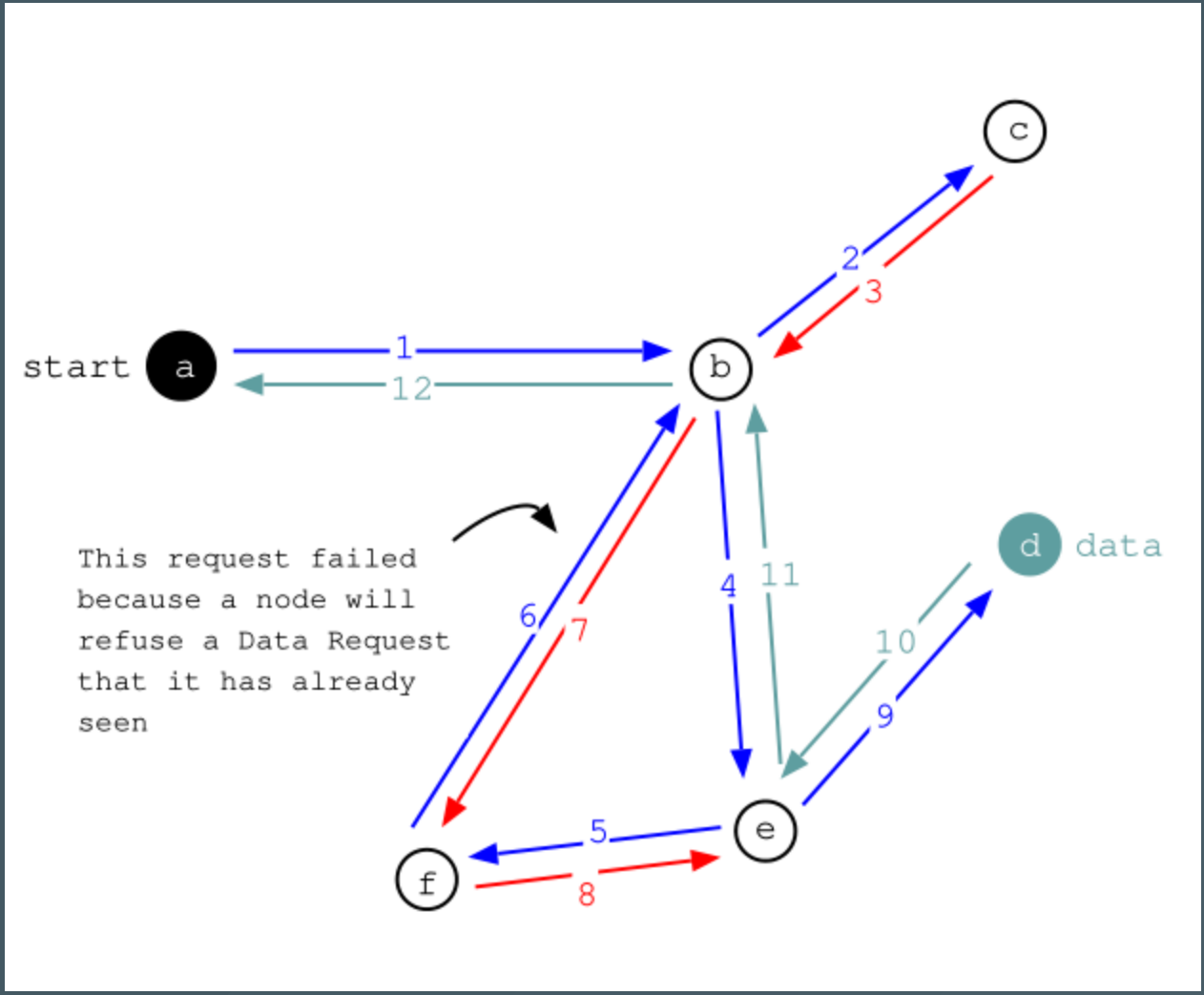
Ottenere un file: Note

- Una richiesta successiva per lo stesso file **può essere soddisfatta dai nodi che hanno fatto caching**
- La forwarding table non deve diventare un elenco di file sorgenti:
  Ogni nodo può decidere **arbitrariamente** di proclamarsi sorgente
- Se un nodo non può soddisfare una richiesta: messaggio di **failure** nella reply, il nodo upstream proverà con il secondo nodo più vicino *[backtracking]*

Ottenere un file: Note

- La qualità del routing **migliora nel tempo**, dato che i nodi si specializzano nel localizzare **set di chiavi simili**, e salvarli in **cluster**
- Il meccanismo **replica in maniera trasparente dati popolari** vicino ai nodi che li richiedono
- Un nodo che risponde con successo a più richieste riesce a scoprire una **maggiore parte della rete** (ma gli altri nodi non lo contattano più facilmente)

Ottenere un file



Small World Network

Opennet e Darknet

Si può discriminare tra 2 tipi di reti P2P:

- **Light**: comunicazione con qualunque nodo
  [Gnutella, Distributed Hash Tables, Opennet]
- **Dark**: comunicazione sono con nodi **fidati** il cui indirizzo è conosciuto a priori.
  [Friend-to-friend network: Waste, Darknet]

Freenet è diviso in **Opennet** e **Darknet**, implementazioni light P2P e dark P2P del protocollo.

Il fenomeno del mondo piccolo

Identifica un network i cui nodi condividono le informazioni **solo con altri nodi conosciuti**. *[fidati]*

Simile al principio dei **sei gradi di separazione** tra esseri umani, ma:

- In una rete di nodi la conoscenza è tradotta come vicinanza o similarità *[closeness]*.

In uno Small World Network, peer simili hanno maggiore probabilità di essere connessi.

SWN Routing

- Peer vicini hanno maggiore probabilità di essere connessi
- Applicazione del **Greedy Routing:** è possibile comunicare con qualunque peer mandando i messaggi **al peer più vicino**, ogni step.

Questo principio è utilizzato sia da DHT che da Freenet per fare routing dei pacchetti.

Il principio di Kleinberg

L'efficacia del routing negli Small World Network dipende dalla proporzione tra le connessioni che hanno diversa lunghezza e la posizione dei nodi. [John Kleinberg, 2000]

Significa:
Il numero di connessioni con determinata lunghezza deve essere inversamente proporzionale alla lunghezza delle stesse.

Permette:
Complessità dell'algoritmo greedy $O(\log^2 N)$

Posizione e Closeness

La posizione dei nodi **non può essere determinata direttamente** in un network anonimo come Darknet.

- Non si conoscono gli **indirizzi** dei nodi
- Non si conoscono le ***trusted connections*** dei nodi
- Le informazioni contenute dai nodi sono distribuite **caoticamente**, non essendoci un'autorità che assegna gli indirizzi.

Come si determina **quale nodo è più vicino al peer che devo contattare?**

Ottimizzazione del Network

- Assegnare ID numerici **unici** e **casuali** ai nodi
- Posizionarli **casualmente** in un grafo, connettendoli alle loro ***trusted connections***

È un **reverse engineering** delle posizioni dei nodi in base alla loro connessioni. Fatto ciò:

- Ogni nodo **scambia la propria posizione con gli altri**, di modo da **minimizzare il prodotto delle lunghezze** degli edge del grafo *[connessioni]*

La rete determina la miglior posizione dei nodi.

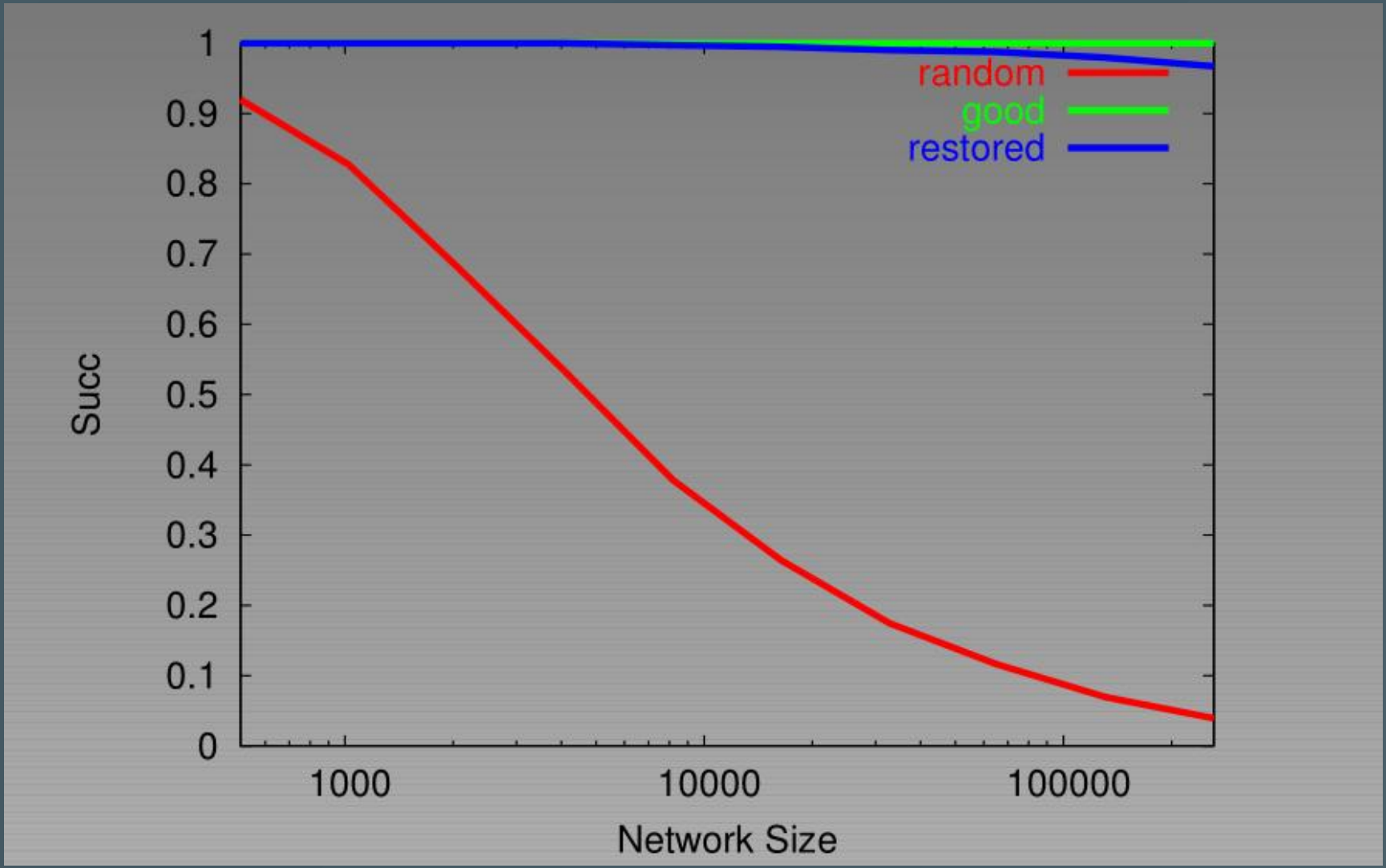
SWN: Conclusioni

- La rete ottimizza le posizioni dei nodi di modo da minimizzare la lunghezza delle connessioni
- La rete così generata permette di raggiungere **qualunque peer** in un numero ragionevole di steps
- La **vicinanza** tra peer è il **prodotto della lunghezza degli edge del grafo**
- L'algoritmo di routing Greedy funziona **se e solo se i nodi vicini hanno più probabilità di conoscersi**. Questo rende l'ottimizzazione del network necessaria.

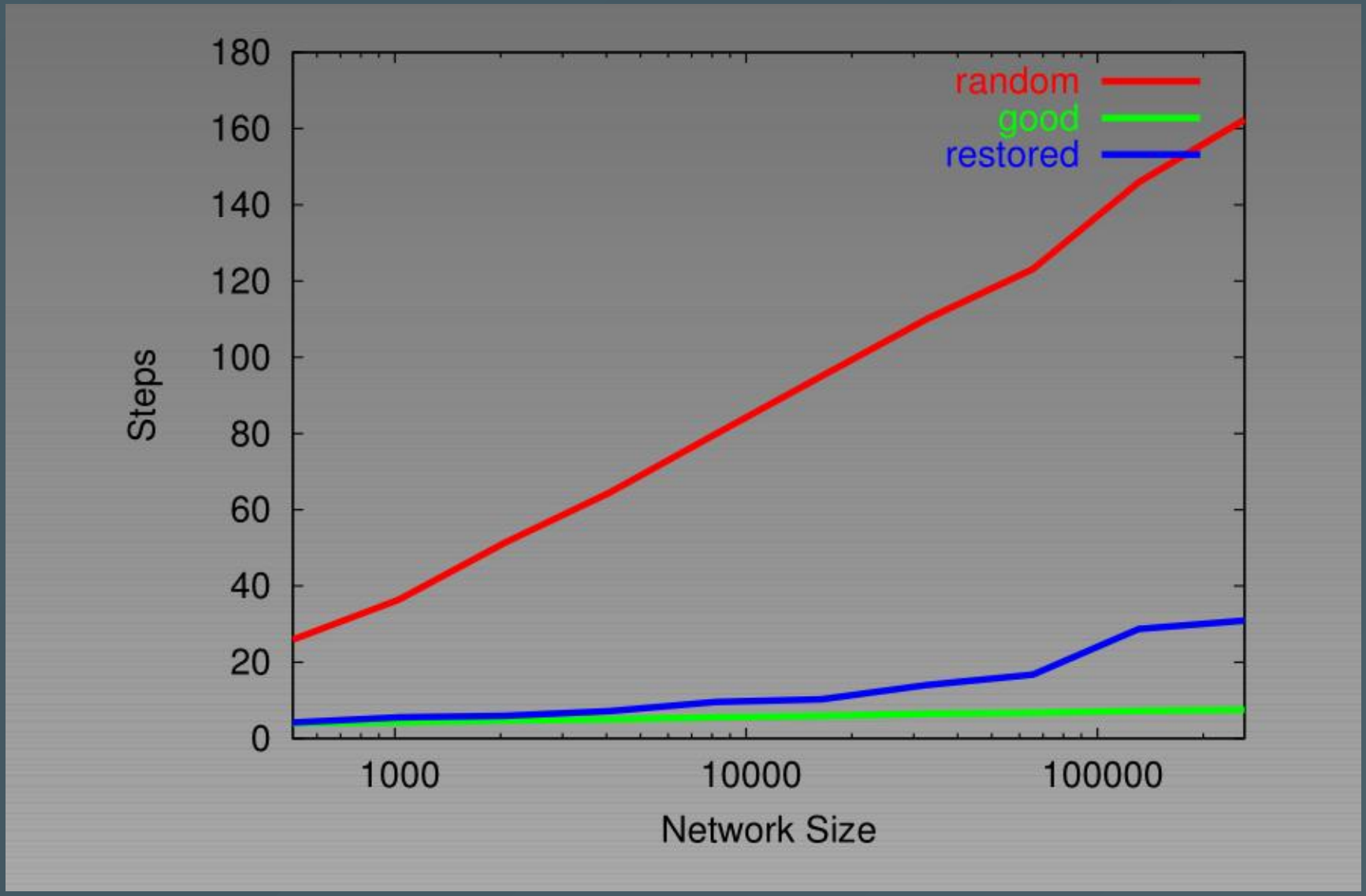
SWN: Simulazione

Si è simulata uno Small World Network con 3 algoritmi di ricerca:

- **Random** tra i nodi *[Random]*
- **Ideal Greedy Routing** (Kleinberg), i nodi sono già disposti in maniera ottima *[Good]*
- **Restored Greedy Routing**, i nodi sono riordinati da Darknet (2000 iterazioni per nodo). *[Restored]*



*Numero di ricerche **successful** in* $log_2^2 N$ *steps.*



*La lunghezza media delle **successful routes**.*

Conclusioni

Comparison di protocolli P2P

Freenet non è l'unico protocollo che tenta di proteggere l'anonimato dei suoi utenti.

A livello di userbase e maturità di progetto, si notano:

- **Tor**, basato sull'**Onion routing**
- **I2P**, basato sul **Garlic routing**
  (variante dell'Onion routing)

Entrambi si identificano piu' come **proxy** che come rete decentralizzata, a differenza di Freenet.

Freenet e Tor

	Freenet	Tor
Protocollo decentralizzato	Sì	No*
Accesso a Internet (proxy)	No	Sì
Accesso a webserver anonimi	No	Sì
Salvataggio distribuito dei dati	Sì	No
Impossibilità di rimuovere i dati	Sì	No
È stato bloccato con successo	No**	Sì

* Tor necessita di una *directory authority* che gestisca gli *hidden services [webserver anonimi].*
** Relativo a Darknet

Grazie per l'attenzione