HS19

DISTRIBUTED SYSTEMS ADVANCED & BLOCKCHAIN

Practical TomP2P

T. Bocek

thomas.bocek@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Distributed Systems - In the *News*

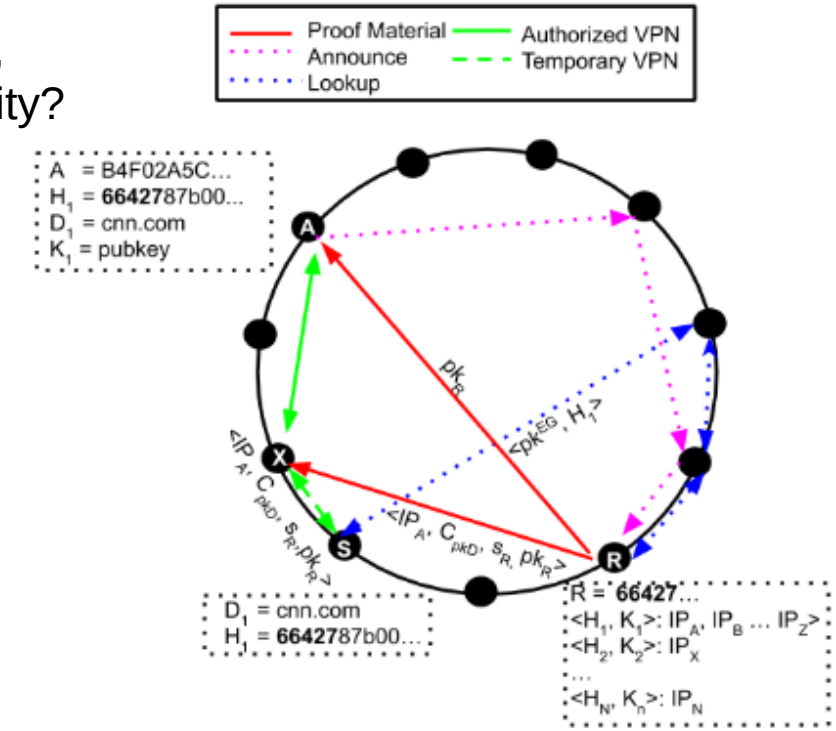
- **07.10.2019: Challenge Task Input: Build Youtube using IPFS**
  - Store videos in IPFS (DHT)
  - I have not tried it, but:
    - “The page may take a minute to load”
- **07.10.2019: Sibex competitor: McAfee unleashes decentralized “no restrictions” crypto exchange**
  - No cross chain atomic swaps, only ERC20
  - Attention its John McAfee: “BTC will be $1 million per coin by 2020”

- **Reminder: Fallacies of distributed computing**
  - The network is reliable
  - Latency is zero
  - Bandwidth is infinite
  - The network is secure
  - Topology doesn't change
  - There is one administrator
  - Transport cost is zero
  - The network is homogeneous

Distributed Systems - In the News

- **02.10.2019: VPN⁰: A Privacy-Preserving Distributed Virtual Private Network**
  - Distributed virtual private network (dVPN)
  - VPN with no central authority (similar to TOR)
    - dVPN: users are both VPN clients and relay/exit nodes
  - Whitelist on exit nodes: traffic accountability
    - Use ZKP to show a given SNI matches my whitelist, but don't reveal which one
  - Whitelist stored in DHT, key is host, value is peer information
  - 10-30 seconds per HTTPS connection (due to slow DHT and ZKP)
  - "S does not include its IP address to the request but rather X's IP so that the destination does not learn which domain S wants to visit." [paper]

- **Discussion**
  - SNI: allows a server to present multiple certificates on the same IP:port
    - IP:port is known, privacy?
  - 1000 white listed domains? – if whitelist is small, privacy?
  - Performance?
  - Network, complexity?



Introduction

TomP2P

A P2P-based high performance key-value pair storage library

- **TomP2P is a P2P framework/library**
  - Implements DHT (structured), broadcasts ([un]structured), direct messages (can implement super-peers)
  - NAT handling: UPNP, NATPMP, relays, hole punching (work in progress)
  - Direct / indirect (tracker / mesh) storage
  - Direct / indirect replication (churn prediction and ~rsync)
  - Modes: key,value / multi-key (versioned) value
  - Java 6, Gradle, Github, Netty, TCP/UDP, MapDB, Android – GoP2P work in progress
- **TomP2P extends DHT**
  - Distributed hash table concept → put(key,value) / get(key)
  - Extended DHT operations →
    - put(key1,key2,value)
    - put prepare / put confirm
    - add(value)
    - digest(key) / bloomfilters / versions
    - get(key) + bloomfilters

Introduction

TomP2P

A P2P-based high performance key-value pair storage library

- **TomP2P started in 24.05.2004**
  - TomP2P v1: Created in 2004 and used for a distributed DNS project
    - This version used blocking IO operations (1 thread / socket)
  - TomP2P v2: Apache MINA (java.nio framework) / 6K LoC
    - Not well designed for non-blocking operations (event-driven)
  - TomP2P v3: Redesigned for non-blocking operations
    - Switched to Netty / 14K LoC, 6K LoC JUnits
  - TomP2P v4: API refinements, new features
    - Latest feature (work in progress) MapReduce
    - 19K LoC (core), 6K LoC JUnits (core)
  - TomP2P v5 (core 18K LoC): modularization, relays, API refinements
- **TomP2P project site**
  - Github
  - TomP2P website (although documentation can be improved)
  - 10 contributors on github, I'm the main one ☺
  - Don't buy the book!

Introduction

TomP2P

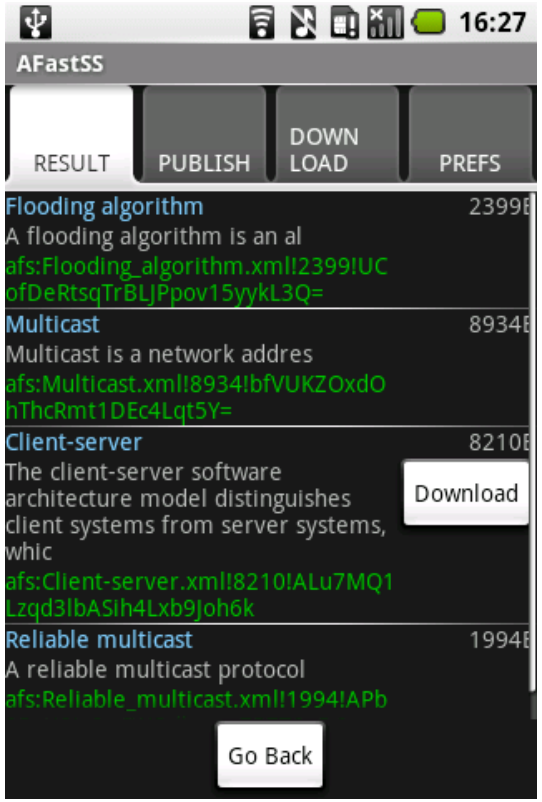
A P2P-based high performance key-value pair storage library

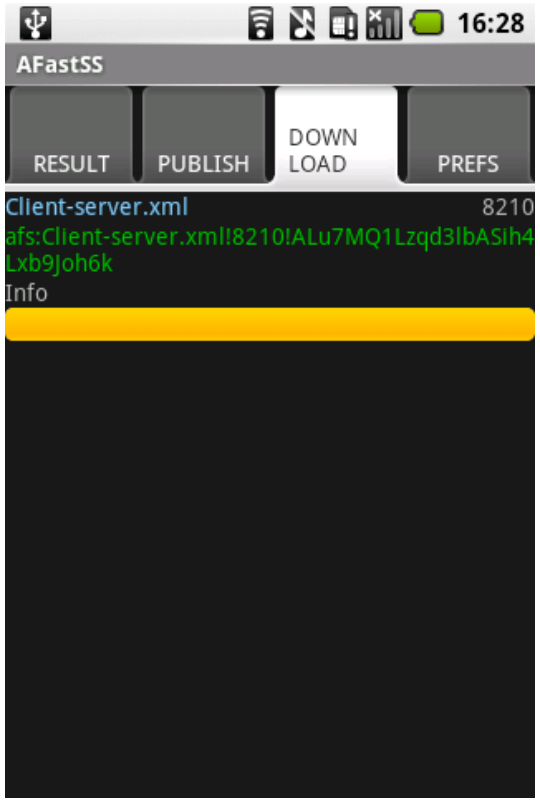
- **Academic background:**
  - Used in EU projects: EC-GIN, Emanics, SmoothIT, SmartenIT, Flamingo
  - Used in research projects: LiveShift, DRFS, Radiommender, Box2Box, Hive2Hive, B-Tracker, PiCsMu, HSR CT 2018
- **http://tomp2p.net**
  - github issues, not in a good state… examples! Use latest release!
  - Documentation: http://tomp2p.net/doc/ Overview: http://en.wikipedia.org/wiki/TomP2P
    - If something is missing, ask
    - Development: https://github.com/tomp2p
    - Feature request possible if good reasons provided

- A Declarative Interface for Smart-phone Based Sensor Network Systems, Asanka Sayakkara and Kasun De Zoysa, IWMS 2012, Bejing, China
- Hybrid Peer-to-Peer DNS, Ricardo Sancho and Ricardo Lopes Pereira, Instituto Superior Tecnico, Porto, Portugal
- A Semantic Publish-Subscribe Coordination Framework for IHE based Cross-Community Health Record Exchange, Visara Urovi, Alex C. Olivieri, Stefano Bromuri, Nicoletta Fornara, Michael Schumacher, ACM SIGAPP Applied Computing Review, 2013 (slides)
- Adding Cryptographically Enforced Permissions to Fully Decentralized File Systems – TUM, Bernhard Amann, Thomas Fuhrmann, April 2011
- Optimis FP7 IP project: Optimized Infrastructure Services, D1.2.1.3, Architecture Design document, ended May 2013
- Distributed Name-based Entity Search, Fausto Giunchiglia and Alethia Hume, ISWC 2012, Boston
- A Distributed Directory System, Fausto Giunchiglia and Alethia Hume, SSWS 2013, Sydney
- A P2P Semantic Query Framework for the Internet of Things, Richard Mietz, Sven Groppe, Oliver Kleine, Daniel Bimschas, Stefan Fischer, Kay Römer and Dennis Pfisterer, PIK, Volume 36, Issue 2 (May 2013)
- P2P Minecraft: "The mods described below are about adding peer-to-peer functionalities to Minecraft."
- Bitcoin Gateway - A Peer-to-peer Bitcoin Vault and Payment Network, Omar Syed & Aamir Syed, July 2011

Introduction

- **TomP2P with Android (early research)**
  - Early adopter
  - TomP2P 5 and Android: ~works





URL: b.socrative.com/student/

Room: HSR

Demo: how to setup TomP2P with IntelliJ

- **build.gradle (on the right)**
- **Simple put/get (Java)**

```
PeerDHT peer1 = new PeerBuilderDHT(new
PeerBuilder(Number160.createHash("test1")).ports(40
00).start()).start();
        PeerDHT peer2 = new PeerBuilderDHT(new
PeerBuilder(Number160.createHash("test2")).ports(40
01).start()).start();

peer1.peer().bootstrap().peerAddress(peer2.peerAddr
ess()).start().awaitListeners();
        peer1.put(Number160.ONE).data(new
Data("hallo")).start().awaitListeners();
        Object
obj=peer2.get(Number160.ONE).start().awaitUninterru
ptibly().data().object();
        System.out.println("out: "+obj);
```

```
plugins {
    id 'java'
}

group 'ch.hsr'
version '1.0-SNAPSHOT'

sourceCompatibility = 1.8

repositories {
    mavenCentral()
        maven {
            url "http://tomp2p.net/dev/mvn/"
        }
}

dependencies {
    testCompile group: 'junit', name: 'junit', version:
'4.12'
    compile group: 'net.tomp2p' , name: 'tomp2p-all',
version: '5.0-Beta8'
}
```

URL: b.socrative.com/student/

Room: HSR

Example

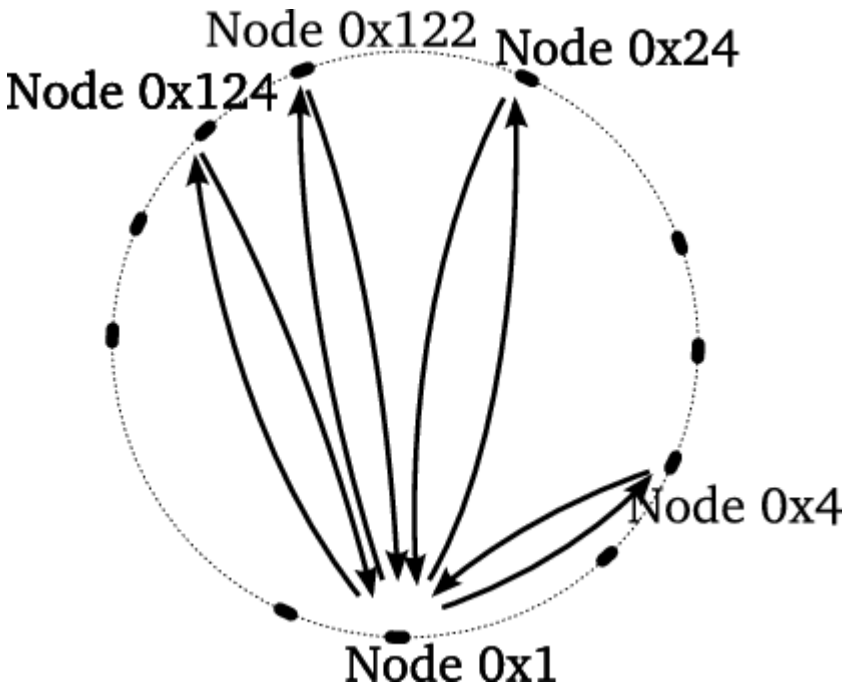
- **Demo: a simple put / get example (example)**
- **Package net.tomp2p.examples. ExamplePutGet**

```
private static void examplePutGet(final PeerDHT[] peers, final Number160 nr)
        throws IOException, ClassNotFoundException {
    FuturePut futurePut = peers[PEER_NR_1].put(nr).data(new Data("hallo")).start();
    futurePut.awaitUninterruptibly();
    System.out.println("peer " + PEER_NR_1 + " stored [key: " + nr + ", value: \"hallo\"]");
    FutureGet futureGet = peers[PEER_NR_2].get(nr).start();
    futureGet.awaitUninterruptibly();
    System.out.println("peer " + PEER_NR_2 + " got: \"" + futureGet.data().object() + "\" for the key " + nr);
    // the output should look like this:
    // peer 30 stored [key: 0xba419d350dfe8af7aee7bbe10c45c0284f083ce4, value: "hallo"]
    // peer 77 got: "hallo" for the key 0xba419d350dfe8af7aee7bbe10c45c0284f083ce4
}
```

- **Defaults**
  - Replication factor 6, replication not enabled,
  - domain, content, version are zero if not specified

Fundamental Concepts (repetition)

- **TomP2P: iterative XOR-based routing**
  - Node and data item unique 160bit identifier
  - Keys are located on the nodes whose node ID is closest to the key



URL: b.socrative.com/student/

Room: HSR

Fundamental Concepts

- **Distributed operations use futures (~promises, Guava)**
- **Future objects**
  - Keeps track of future events, while the “normal” program flow continues → `addListener()` or `await()`
  - `await()`: Operations are executed in same thread
  - `addListener()`: Operations are executed in same or other thread
- **Demo: blocking operation (net.tomp2p.examples. ExamplePutGet)**

```
private static void exampleGetBlocking(final PeerDHT[] peers, final Number160 nr)
    throws ClassNotFoundException, IOException {
    FutureGet futureGet = peers[PEER_NR_2].get(nr).start();
    // blocking operation
    futureGet.awaitUninterruptibly();
    System.out.println("result blocking: " + futureGet.data().object());
    System.out.println("this may *not* happen before printing the result");
}
```

- **Demo: non - blocking operation (net.tomp2p.examples. ExamplePutGet)**
  - Advise: use `addListener(…)` as much as possible!
  - `operationComplete(…)` must be **always** called (problem if not)

```
private static void exampleGetNonBlocking(final PeerDHT[] peers, final Number160 nr) {
    FutureGet futureGet = peers[PEER_NR_2].get(nr).start();
    // non-blocking operation
    futureGet.addListener(new BaseFutureAdapter<FutureGet>() {
        @Override
        public void operationComplete(FutureGet future) throws Exception {
            System.out.println("result non-blocking: " + future.data().object());
        }

    });
    System.out.println("this may happen before printing the result");
}
```

  - New utilities necessary (loops as recursions)

Fundamental Concepts

- **Future utilities**
  - `FutureForkJoin(int nr, boolean cancel, K... Forks)`
    - Joins already “forked” futures. Waits until all or `nr` future finished. If nr reached, futures may be cancelled (e.g. abort download)
  - `FutureLateJoin(int nrMaxFutures, int minSuccess)`
    `FutureLaterJoin()`
    - No need to add the futures in the constructor, can be added later
  - `FutureDone()`
    - A generic future used in many places, can be placeholder
- **Needs face-lifting, Java8 with CompletableFuture**
- **C# / JavaScript and other languages have better support for async**
  - Example:

```
public async Task MyMethod()
{
    Task<int> longRunningTask = LongRunningOperation();
    //indeed you can do independent to the int result work here

    //and now we call await on the task
    int result = await longRunningTask;
    //use the result
    Console.WriteLine(result);
}

public async Task<int> LongRunningOperation() // assume we return an int from thi
{
    await Task.Delay(1000); //1 seconds delay
    return 1;
}
```

Fundamental Concepts

- **API Overview: Peer.java**
  - Core methods, network related
    - `sendDirect()`
    - `bootstrap()`
    - `announceShutdown()`
    - `ping()`
    - `discover()`
    - `broadcast()`
  - Methods for DHTs (PeerDHT.java)
    - `put(key, value)`
    - `get(key)`
    - `add(value)`
    - `digest(key)`
    - `remove(key)`
    - `send(key)`
    - `parallelRequest(key) // mostly used internally`

- **Extensions**
  - TomP2P can store multiple values for a key
    - `put() (location_key, content_key, value)` → content_key specified in Builder
    - `get().all()` → returns a map with [`content_key, value`]
    - `add() (location_key, value)` → is translated to `put() (location_key, hash(value), value)`
  - TomP2P support domains
    - Avoid collision for same keys
    - Domains are used for protection
    - Domains specified in Builder
    - `put() (key, domain, value)` → `get() (key,domain)`

Connectivity, Security, and Robustness

- **Security in TomP2P (best-effort security)**
  - Signature-based, no data encryption
  - Messages are signed using SHA1 with DSA
    - Too weak, don't use it!
  - Sybil attacks!
    - This attack creates large number of identities, may collude
- **How to prevent Data from being overwritten**
  - Domain and entry protection, requires cooperation
  - `StorageLayer.protectionDomainMode(…)`

For domains and entries		
protectionEnabled	ALL	NONE
protectionMode	NO_MASTER	MASTER_PUBLIC_KEY

- **Domain / entry protection**
  - Set publick key `new PeerBuilder(PublicKey)`
    - Enable=ALL, Mode=NO_MASTER → every peer can protect domains, first come first served
    - Enable=NONE, Mode=NO_MASTER → no peer can protect domains
    - Enable=ALL, Mode=MASTER_PUBLIC_KEY → every peer can protect domains, the owner can claim domain
    - Enable=NONE, Mode=MASTER_PUBLIC_KEY → no peer can protect domains except the owner
  - Owner of domain 0x1234 is peer where 0x1234 == hash(public_key)

URL: b.socrative.com/student/

Room: HSR

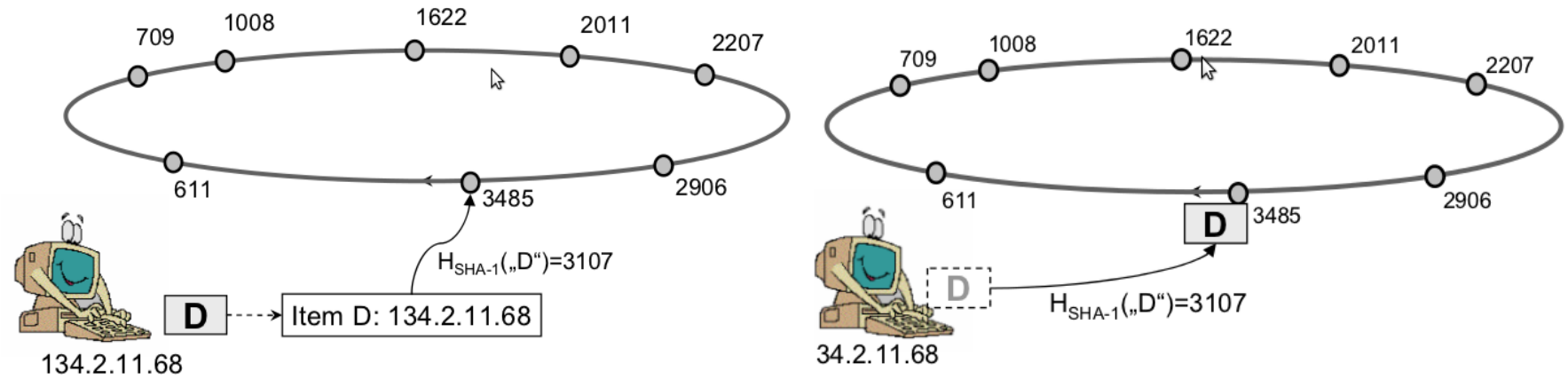
From the VSS Exercises

- **Attacking the DHT**
  - Create a key for a data item close to the target:
    Number160.createHash(data).xor(new Number160(0)) – distance 0, perfect match
    Number160.createHash(data).xor(new Number160(1)) – distance 1
    Number160.createHash(data).xor(new Number160(2)) – distance 2
    …
  - Or create key of node close to the target
    new PeerBuilder(new Number160(RND)).ports(port).start(), where RND is
    Number160.createHash(data).xor(new Number160(0))
    Number160.createHash(data).xor(new Number160(1))
    …
  - Peer can then answer there is no data
  - For previously known values / peers (known public key)
    - Cannot change data, but make it disappear

Components with Examples (repetition)

- **DHT vs. Tracker**
  - DHT “stored by value” – direct storage
  - Tracker “stored by reference” – indirect storage

indirect (Tracker) **direct (DHT)**



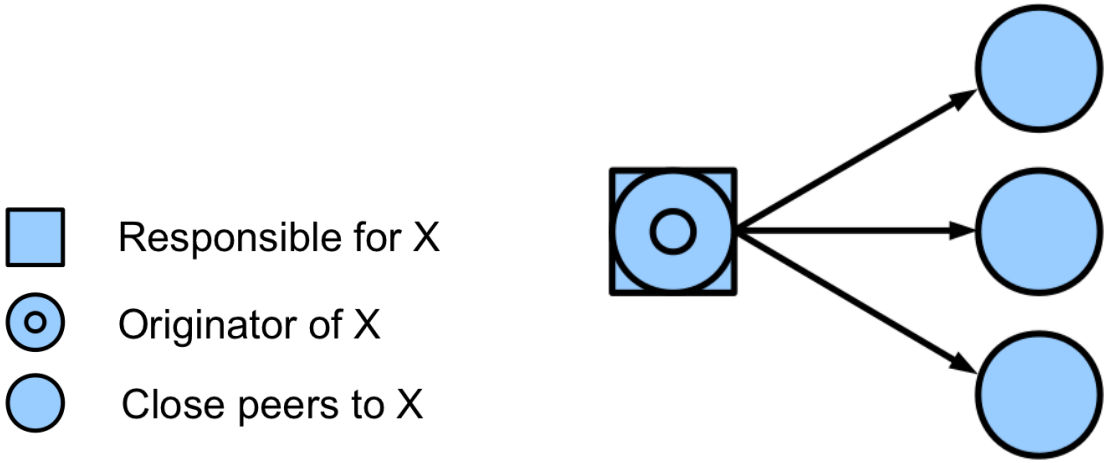
Components with Examples

- **Demo: Tracker (net.tomp2p.examples.ExampleTracker)**
  - Create 100 peers,
  - Add to tracker, get from tracker
  - Stored on 3 peers: TrackerBuilder.java (can be configured)
  - Attachment of data is possible (attachement(Data))

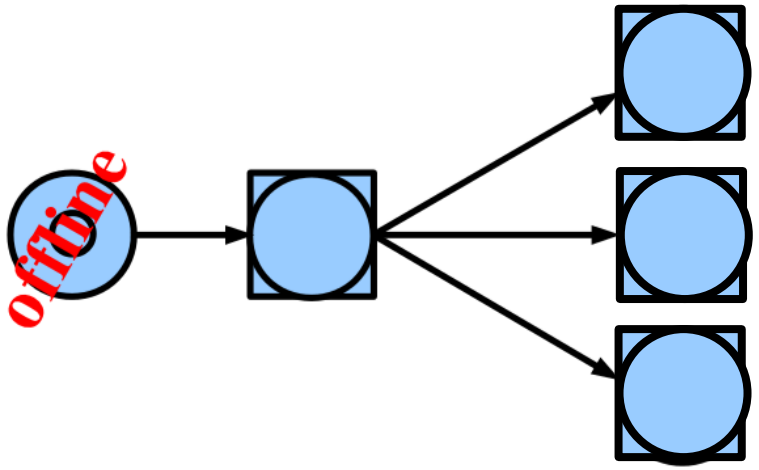
```
private static void example(final PeerTracker[] peers) throws IOException, ClassNotFoundException {

    FutureTracker futureTracker = peers[12].addTracker(Number160.createHash("song1")).start().awaitUninterruptibly();
    System.out.println("added myself to the tracker with location [song1]: "+futureTracker.isSuccess()+" I'm: "+peers[12].peerAddress());

    FutureTracker futureTracker2 = peers[24].getTracker(Number160.createHash("song1")).start().awaitUninterruptibly();

    System.out.println("peer24 got this: "+futureTracker2.trackers());
    System.out.println("currently stored on: "+futureTracker2.trackerPeers());
}
```

Connectivity, Security, and Robustness

- **Direct Replication**
  - Replication: enough replicas
    - Originator peer is responsible
    - Periodically refresh replicas
    - Example: tracker that announces its data
  - Originator offline → replicas disappear.



- **Indirect Replication**
  - The closest peer is responsible, originator may go offline
  - Periodically checks if enough replicas exist
    - Detects if responsibility changes
    - Requires cooperation between responsible peer and originator
    - Multiple peers may think they are responsible for different versions → eventually solved



nRoot (default is 0Root)

Components with Examples

```
new IndirectReplication(peer1).start();
new IndirectReplication(peer2).start();
new IndirectReplication(peer3).start();
```

- **Demo: Direct Replication (net.tomp2p.examples.ExampleDirectReplication)**
  - Create 100 peers
  - Create put builder
  - Create JobScheduler with that builder
  - Put: repeat forever
  - Remove: try 9 times

- **Demo: Direct Replication (net.tomp2p.examples.ExampleIndirectReplication)**
  - Create 100 peers, store data on one peer
  - Setup indirect replication
  - Bootstrap peer 1 to peer 2, transfer data
  - Bootstrap peer 1 to peer 3, now peer 3 is closest and responsible
  - Change bootstrap order!

```
private static void exmpleDirectReplication(final PeerDHT[] peers) throws IOException, InterruptedException {
    PutBuilder putBuilder = peers[1].put(Number160.ONE).data(new Data( object: "test"));
    JobScheduler replication = new JobScheduler(peers[1].peer());
    Shutdown shutdown = replication.start(putBuilder, intervalMillis: 1000, repetitions: -1, (future) → {
            System.out.println("put again...");
    });
    Thread.sleep(NINE_SECONDS);
    System.out.println("stop replication");
    shutdown.shutdown();
    RemoveBuilder removeBuilder = peers[1].remove(Number160.ONE);
    replication.start(removeBuilder, intervalMillis: 1000, repetitions: 9, (future) → { System.out.println("remove again..."); });
    Thread.sleep(NINE_SECONDS);
    System.out.println("done");
    replication.shutdown().awaitUninterruptibly();
}
```

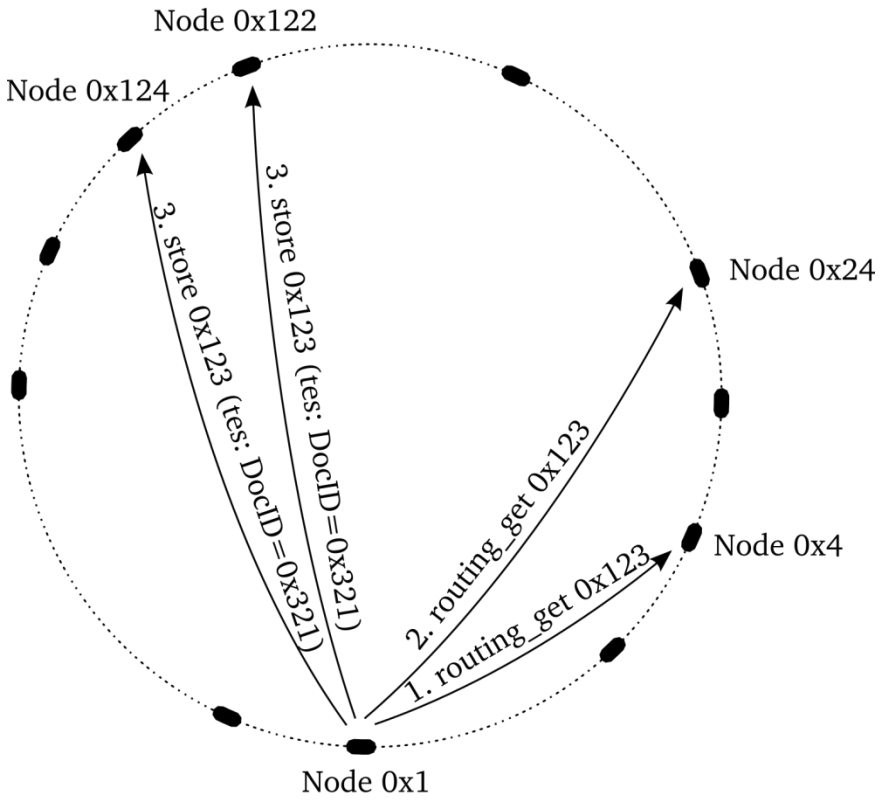
Mechanisms based on Hashing in DHTs (VSS repetition)

- **Mechanism that brings similarity search to HT / DHT**
  - Problem: Search for “netwrk” fails for DHTs
- **Similarity: Edit distance / Levenshtein distance**
  - Min operations to transform one string into another, operations: insert, delete, replace
  - Calculated in matrix size O(m x n)
- **Example d(test,east) = 2 (remove a, insert t)**
- **Main idea: pre-calculate errors**
  - All possible errors? Neighbors for test with ed 2: test, testa, testaa, testab, ... , tea, teb, tec, ..., teaa, teab, ... → 23883 more of those!
- **FastSS pre-calculates with deletions only**
  - Neighbors for *test* with ed 1: test, est, tst, tes

$$d[i,0] = i,\ d[0,j] = j,$$
$$d[i,j] = min\,(d[i-1,j]+1,\ d[i,j-1]+1,$$
$$d[i-1,j-1] + (if\ s1[i] = s2[j]\ then\ 0\ else\ 1))$$

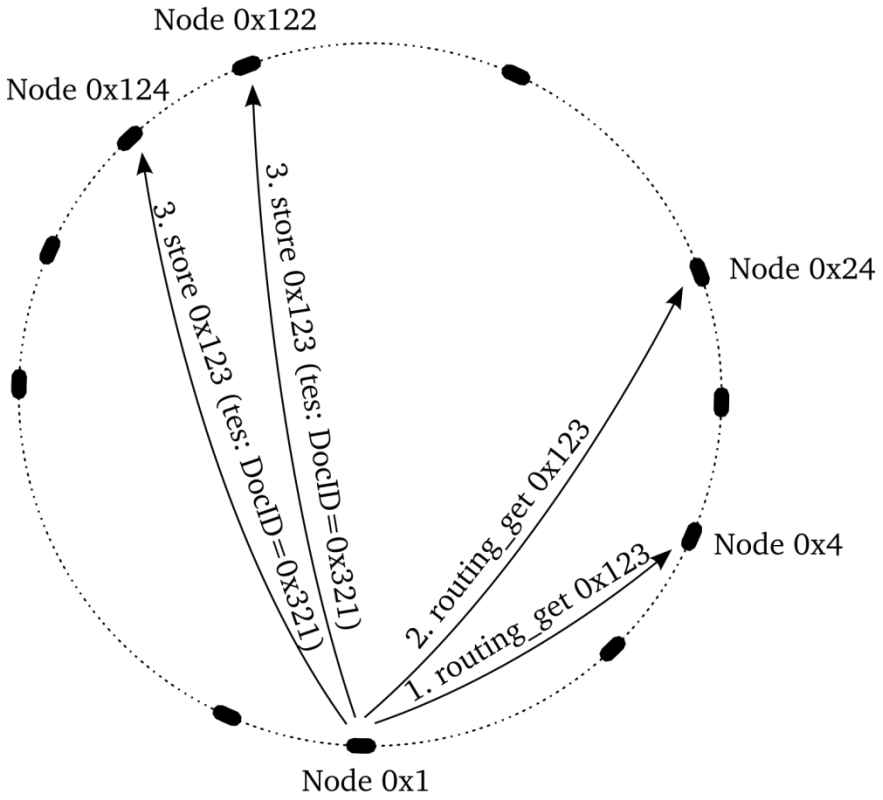
Mechanisms based on Hashing in DHTs

- **Index documents using put(hash(document), document)**
  - Document (0x321) contains word test
- **Index all neighbors (test, tes, tst, tet, est) using put(hash(neighbor), point to document)**
  - hash(“tes”) = 0x123



Mechanisms based on Hashing in DHTs

- **User searches for “tesx”**
- **Neighbors are generated (tesx, esx, tsx, tex, tes)**
  - get(hash(neighbor)) → 0x123
  - Find pointer to document (0x321)
  - document = get(0x321)
- **Tests with edit distance 1, partially 2, ignoring delete pos.**
  - Overhead (*n choose k*) for query and index
- **Similarity search as series of `put()` and `get()`**



Components with Examples

- **Demo: Similarity Search (net.tomp2p.examples.ExampleFastSS**
  - 100 peers
  - Add hashed title to tracker
  - Create reply handler for that file
  - Index with FastSS keywords
  - Search for misspelled keyword
  - Find key to title
  - Download title

```
        for (String deletion : deletion( word: "greet")) {
            FutureGet futureGet = peers1[20].get(Number160.createHash(deletion)).start().awaitUninterruptibly();
            if (futureGet.isSuccess() && futureGet.data()!=null) {
                // if we found a match
                Object[] tmp = (Object[]) futureGet.data().object();
                Number160 key1 = (Number160) tmp[0];
                // get the peers that have this file
                FutureTracker futureTracker = peers2[20].getTracker(key1).start();
                futureTracker.awaitUninterruptibly();
                PeerAddress peerAddress = futureTracker.trackerData().iterator().next().peerAddresses().keySet()
.iterator().next();
                // download
                FutureDirect futureDirect = peers1[20].peer().sendDirect(peerAddress).object(key1).start();
                futureDirect.awaitUninterruptibly();
                System.out.println("we searched for \"greet\", and found [" + tmp[2] + "], ed(" + tmp[1] + ",greet)="
                        + ld((String) tmp[1], t: "greet") + ". After downloading we get [" + futureDirect.object() +
"]");
            }
        }
```

URL: b.socrative.com/student/

Room: HSR

Questions?

Dr. Thomas Bocek thomas.bocek@hsr.ch
Roman Blum roman.blum@hsr.ch