Lecture 5

P2P with TomP2P

http://tomp2p.net

*Original slides for this lecture provided by David Hausheer (TU Darmstadt, Germany), Thomas Bocek, Burkhard Stiller (University of Zürich, Department of Informatics, Communication Systems Group CSG, Switzerland,

0. Lecture Overview

1. **Introduction**
   1. What is TomP2P
   2. History and project information
2. **Example**
3. **Fundamental concepts**
   1. XOR-based iterative routing
   2. Futures
   3. API Overview
4. **Components with examples**
   1. DHT with examples
   2. Tracker / PEX with examples
5. **Advanced Topics in TomP2P**
   1. Bloom Filters

P2P in the news

- **14.03.2016** Starker Auftritt der Schweizer IT-Industrie an der weltweit grössten Computermesse. Aber auch die Konkurrenz schläft nicht.
  - **Coinblesk @CeBIT**
  - **Currently: 2 MA (Andreas, Alessandro), 1 BA (Christian)**
- **09.03.2016** UZH an der CeBIT 2016: Schnell und sicher mit Bitcoins mobil bezahlen
- **17.03.2016** Blitzschnelle Bitcoins
- **14.03.2016** Ethereum Blockchain Project Launches First Production Release
  - Fundamental change in the protocol makes older versions incompatible → hard fork
  - Bitcoin Classic vs. Core

1. Introduction

What is TomP2P

History and project information

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, new addition: 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, Maven, Github, Netty, TCP/UDP, MapDB, (Android)

Introduction

TomP2P
A P2P-based high performance key-value pair storage library

- **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

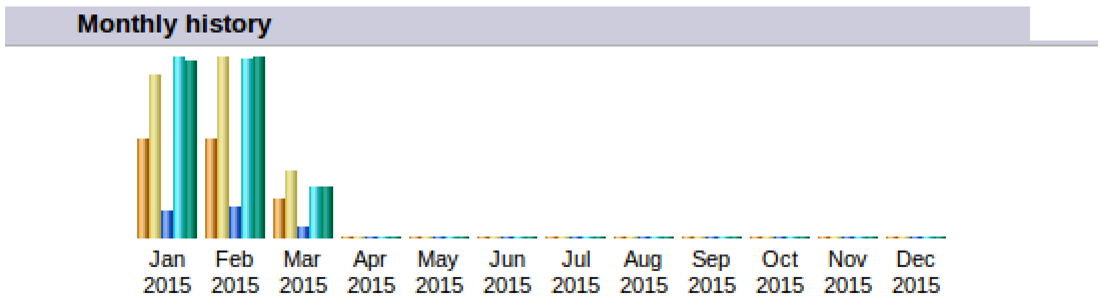
TomP2P
A P2P-based high performance key-value pair storage library

- **TomP2P history**
  - 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

Introduction

TomP2P
A P2P-based high performance key-value pair storage library

- **TomP2P started in 24.05.2004**
  - Github: watch 34 / star 192 / fork 68 (not sure if good/bad)
  - TomP2P website (although documentation is outdated)
  - 75 users on the mailinglist
  - 11 contributors on github
  - Don’t buy the book!



Month	Unique visitors	Number of visits	Pages	Hits	Bandwidth
Jan 2015	1,785	2,943	10,859	71,716	606.09 MB
Feb 2015	1,802	3,254	12,423	71,149	617.73 MB
Mar 2015	721	1,218	4,350	20,055	176.03 MB
Apr 2015	0	0	0	0	0
May 2015	0	0	0	0	0

Introduction

TomP2P
A P2P-based high performance key-value pair storage library

- **Academic background (CSG - UZH):**
  - Used in EU projects: EC-GIN, Emanics, SmoothIT, SmartenIT, Flamingo
  - Used in research projects: LiveShift, DRFS, Radiommender, Box2Box, Hive2Hive, B-Tracker, PiCsMu, peerwasp, (and non-academic)
- **http://tomp2p.net**
  - For questions: mailinglist (http://lists.tomp2p.net/cgi-bin/mailman/listinfo)
  - Specific questions: bocek -at- ifi.uzh.ch or tom -at- tomp2p.net
  - Documentation: http://tomp2p.net/doc/ (TomP2P v4.4)
    Overview: http://en.wikipedia.org/wiki/TomP2P
    - If something is missing, ask! – **Documentation for v5 is missing!**
  - Development: https://github.com/tomp2p
    - Feature request possible if good reasons provided
- **(Demo: how to setup TomP2P with Netbeans/git/maven)**

Introduction

TomP2P
A P2P-based high performance key-value pair storage library

- 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)

Introduction

TomP2P
A P2P-based high performance key-value pair storage library

- 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

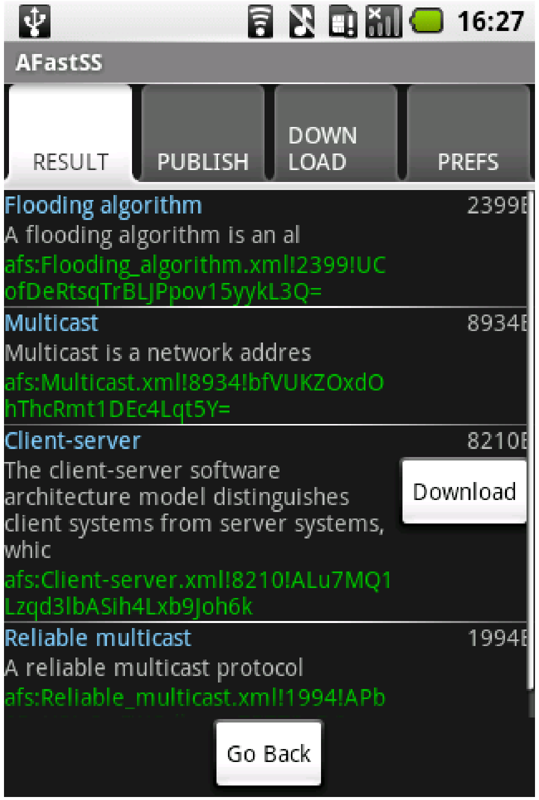
Introduction

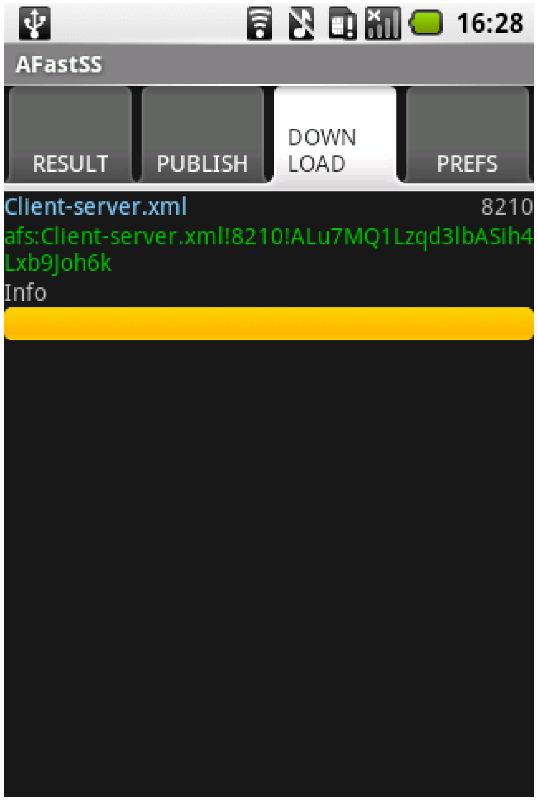
TomP2P
A P2P-based high performance key-value pair storage library

- 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
- Bitsquare.io – The decentralized bitcoin exchange, switched to Tor

Introduction

- **TomP2P with Android (early research)**
  - CSG: early adopter
  - TomP2P 5 and Android: work in progress





2. Example

Example and Demo

Example

- **Demo: a simple put / get 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

3. Fundamental Concepts

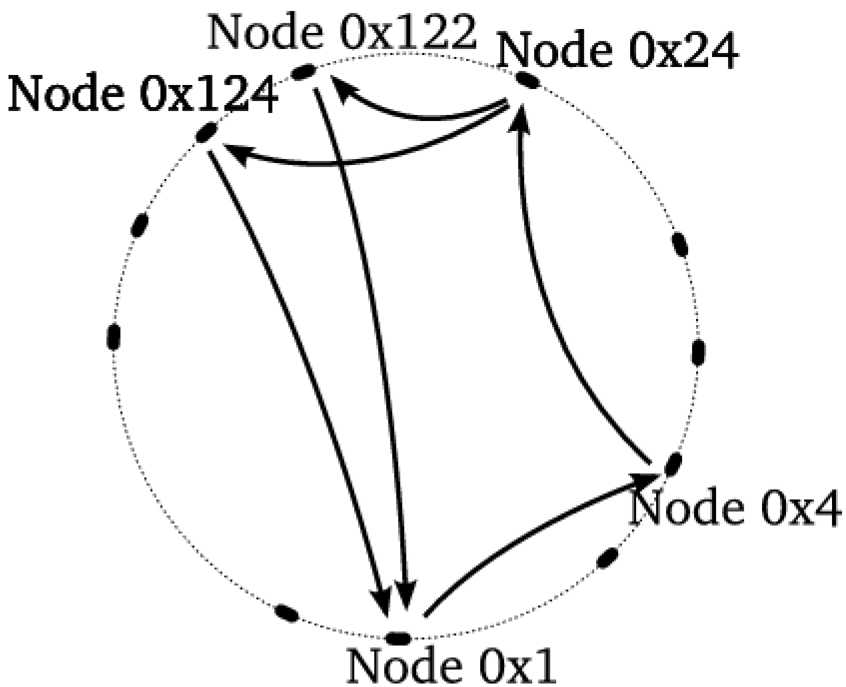
XOR-based iterative routing

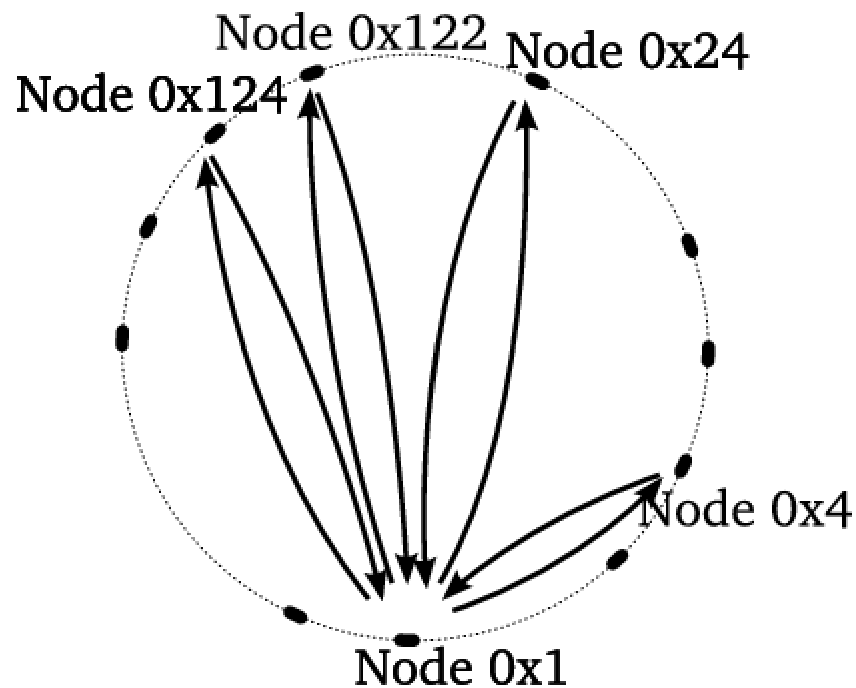
Futures

API Overview

Fundamental Concepts (repetition)

- **Recursive routing vs. iterative routing**





+ online status update

- faulty peers cause delay

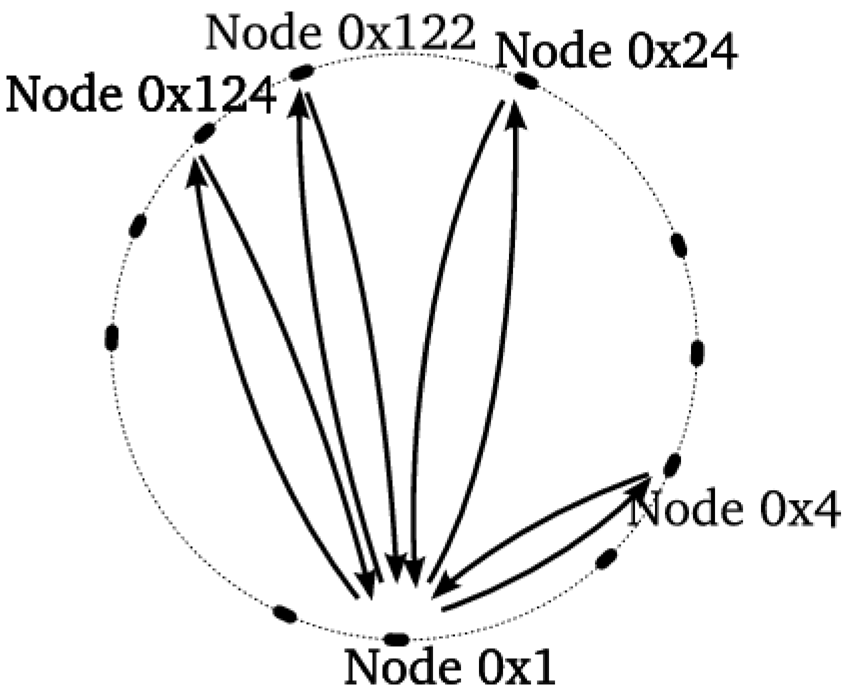
+ control

- neighbor maintenance

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
  - Search for a key:
    - Lookup in neighbor table for closest peer (*e.g.* peers with ID: 0x1, 0x2, 0x3, 0x4)

My ID	Neighbor ID	Distance (XOR)
1	2	3
1	3	2
1	4	5



    - Difference to Pastry: one metric, no leaf set / routing table

Fundamental Concepts

- **TomP2P iterative XOR-based routing**
  - Neighbors stored in 159 “bags”, bag has capacity c (Kademlia, c=20)
  - Routing takes O(log n) → M03, slides 15
  - By default UDP, message header 63 bytes
  - Routing Mechanism variables, can be tuned
    - directHits, potentialHits – routing sends digest
    - forceTCP – use TCP instead of UDP
    - maxDirectHits, maxNoNewInfo, maxSuccess, maxFailure – stop conditions
    - parallel – number of parallel connections
  - For the CT - don’t worry, default settings are fine ☺

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)**

```java
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");
}
```

Fundamental Concepts

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

```java
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");
}
```

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
- **ForkJoin in Java7**
  - Fork and join framework – future utilities in TomP2P focus on join, forking is done “manually”
- **Needs face-lifting, Java8 with [CompletableFuture]()**

Fundamental Concepts

- <u>Fun</u> with futures: loops

```
Future loop() {
		Future future = new Future();
		recLoop(future);
		return future;
}

void recLoop(Future future) {
		int active = 0;
		for (int i = 0; i < parallel; i++) {
			//if future finished, it will be set to null
			if (futureResponses[i] == null) {
				active++;
				futureResponses[i] = doSomething();
			}
			else if (futureResponses[i] != null) active++;
		}
		if (active == 0) future.weAreDone();
		FutureForkJoin<FutureResponse> fp = new FutureForkJoin<FutureResponse>(1, futureResponses);
		fp.addListener(new BaseFutureAdapter<FutureForkJoin<FutureResponse>>() {
			@Override
			public void operationComplete(FutureForkJoin<FutureResponse> future)
			throws Exception {
				boolean finished = evaluate(future);
				if(finished) future.weAreDone();
				else recLoop(future);
			}
		});
}
```

Fundamental Concepts

- **Java 8 lambda expressions not used**
- **.NET 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(key)`
    - `digest(key)`
    - `remove(key)`
    - `send(key)`
    - `parallelRequest(key) // mostly used internally`

Fundamental Concepts

- **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 (more details later)
    - Domains specified in Builder
    - `put() (key, domain, value)` → `get() (key,domain)`

Configurations Example

- Configuration with builder pattern

```
RoutingConfiguration rc = new RoutingConfiguration(0, 0, 1);
RequestP2PConfiguration pc = new RequestP2PConfiguration(1, 0, 0);
FuturePut fdht = peers[444].put(peers[30].peerID()).data(new Number160(5), data)
        .domainKey(Number160.createHash("test")).routingConfiguration(rc)
        .requestP2PConfiguration(pc).start();
```

- System-wide configuration when creating Peer

```
Number160 peerId = new Number160(rnd);
PeerMap peerMap = new PeerMap(new PeerMapConfiguration(peerId));
master = new PeerBuilder(peerId)
        .ports(port).enableMaintenance(maintenance)
        .bindings(bindings).peerMap(peerMap).start().addAutomaticFuture(automaticFuture);
peers[0] = new PeerBuilderDHT(master).start();
```

5. Components with Examples

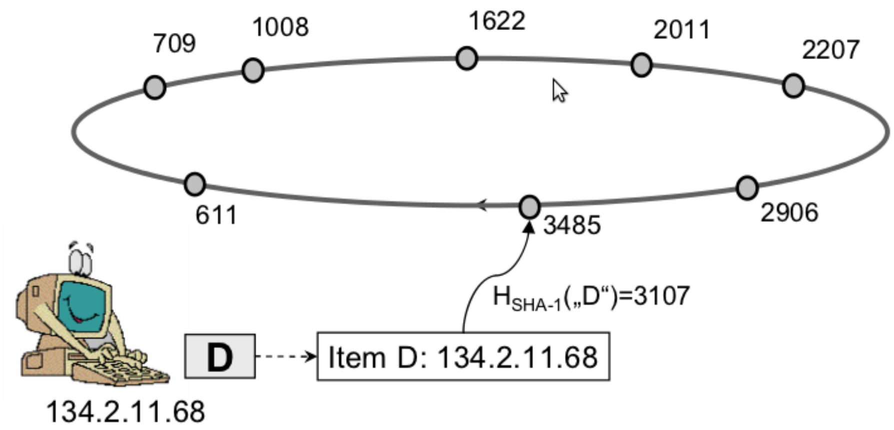
DHT

Tracker

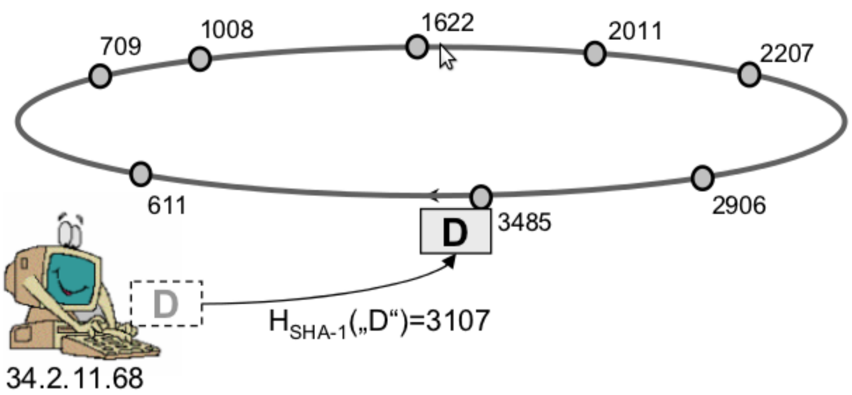
Components with Examples (repetition)

- **DHT vs. Tracker**
  - M03, slide 27: DHT “stored by value” – direct storage
  - M03, slide 28: Tracker “stored by reference” – indirect storage

indirect (Tracker)



direct (DHT)



Components with Examples

- **B-Tracker**
  - Centralized tracker – one machine gets traffic
  - DHT: store reference on 20 peers – 20 peers gets traffic
  - PEX: exchange information every minute (push)
  - B-Tracker, every downloading peer becomes a tracker → forms mesh
    - Better balance of load
    - To avoid duplicates send compressed list of known peers
  - B-Tracker in TomP2P enabled by default

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))

```java
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());
}
```

Components with Examples

- **Demo: Tracker**
  - Although demo uses `await()`, try not to use it
  - Tracker vs. DHT what is better for the CT? You decide!

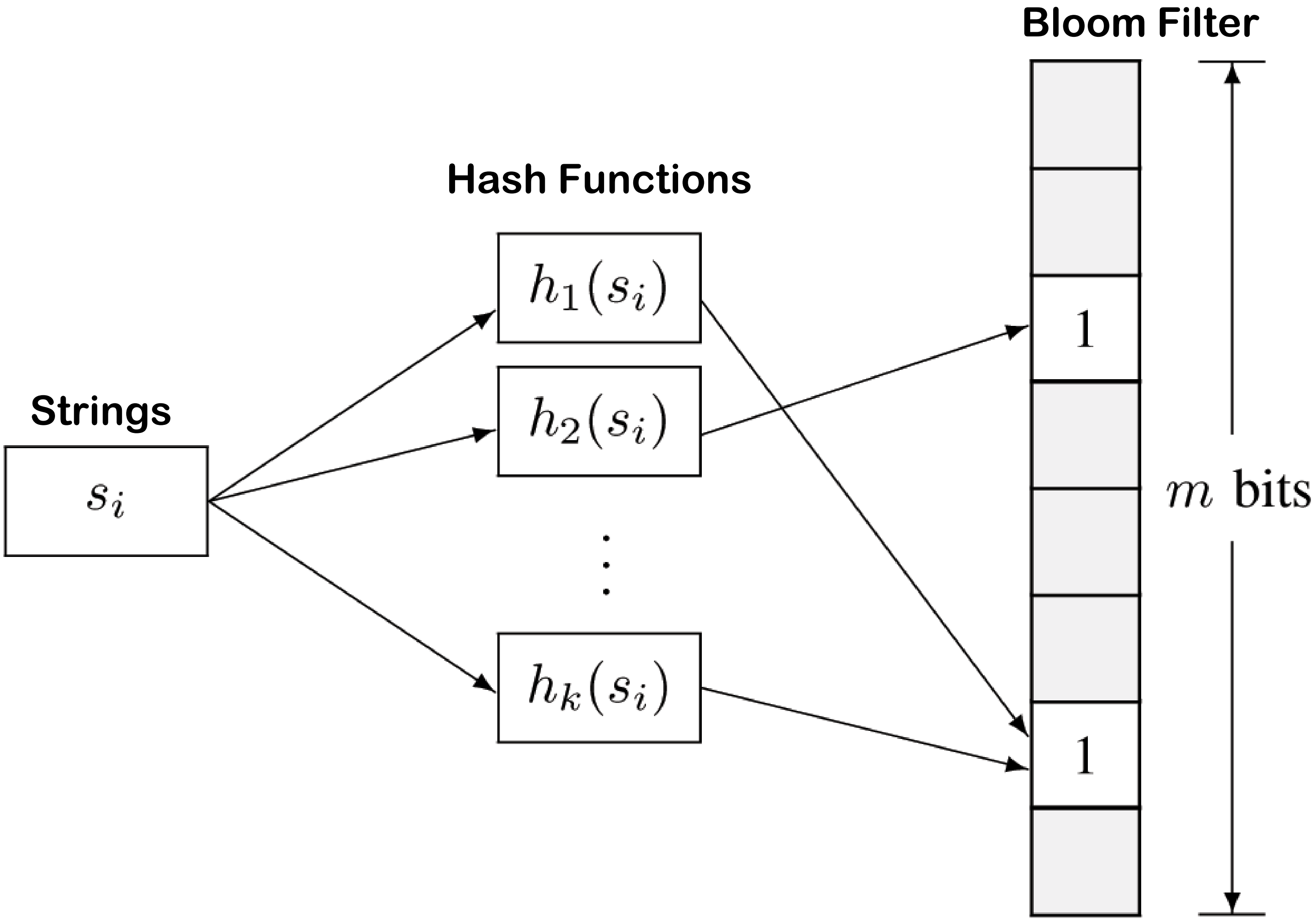
- **Further interesting aspects for the challenge task:**
  - To be discussed on Thursday
  - Reminder: Thursday starts the challenge task
  - Task presentation and Scrum introduction

5. Bloom Filters

Traditional Bloom Filter

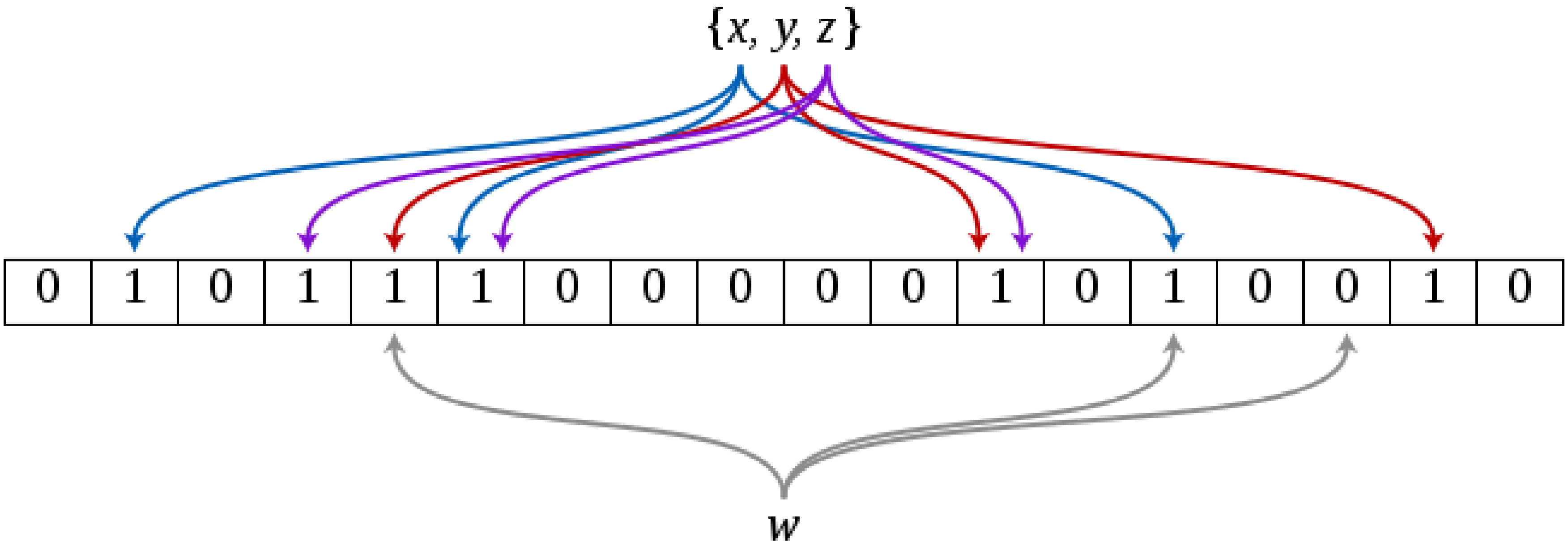
- **An array of *m* bits, initially all bits set to 0**
- **A bloom filter uses *k* independent hash functions**
  - *h1, h2, …, hk* with range {1, …, m}
- **Each key is hashed with every hash function**
  - Set the corresponding bits in the vector
- **Operations**
  - Insertion
    - The bit A[hi(x)] for 1 < i < k are set to 1
  - Query
    - Yes if all of the bits A[hi(x)] are 1, no otherwise
  - Deletion
    - Removing an element from this simple Bloom filter is impossible

Insertion of an Element



Query of an Element, m=18, k=3

- Insert x, y, z
- Query w



http://en.wikipedia.org/wiki/Bloom_filter

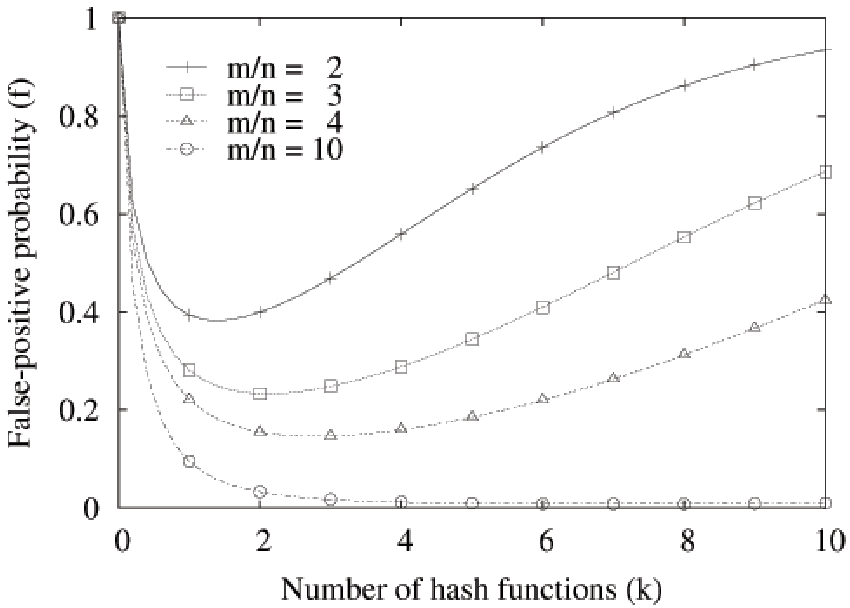
Properties

- **Space Efficiency**
  - Any Bloom filter can represent the entire universe of elements
    - In this case, all bits are 1
- **No Space Constraints**
  - Add never fails
  - But false positive rate increases steadily as elements are added
- **Simple Operations**
  - Union of Bloom filters: bitwise OR
  - Intersection of Bloom filters: bitwise AND

False-Positive Probability

- **No false negative, but false positive**
- **False-positive probability:**
  - *n* number of strings; *k* hash functions; *m*-bit vector

$$f = (1 - e^{-\frac{nk}{m}})^k$$



=> Given m/n, there is an optimal number of hash functions (opt. k = m/n ln 2) (when 50% of the bits are set)

Examples

- **Example for False-positives**
  - Insertions
    - Hash („color printer“) => (1,4,6)
    - Hash („digital camera“) => (3,4,5)
    - Bloom filter (1,3,4,5,6)
  - Query
    - Hash („heat sensor“) => (3,4,6)
    - Matches since bits 3,4,6 are all set to 1
  - Online
- **False-negative**
  - Query
    - Hash (“color printer”) => (1,4,6) , matches (1,3,4,5,6) → no false-negative

Bloom Filter Variants (1)

- Compressed Bloom Filters
  - When the filter is intended to be passed as a message
  - False-positive rate is optimized for the compressed bloom filter (uncompressed bit vector m will be larger but sparser)
  - However, compression/decompression, more memory
- Generalized Bloom Filter
  - Two type of hash functions gi (reset bits to 0) and hj (set bits to 1)
  - Start with an arbitrary vector (bits can be either 0 or 1)
  - In case of collisions between gi and hj, bit is reset to 0
  - Store more bits with low false positive
  - Produces either false positives or false negatives

Bloom Filter Variants (2)

- **Counting Bloom Filters**
  - Entry in the filter not be a single bit but a counter
  - Delete operation possible (decrementing counter)
  - Variable-Increment Counting Bloom Filter
- **Scalable Bloom Filter**
  - Adapt dynamically to number of elements, consist of regular Bloom filters
  - “A SBF is made up of a series of one or more (plain) Bloom Filters; when filters get full due to the limit on the fill ratio, a new one is added; querying is made by testing for the presence in each filter”

Example and Applications

- **Demo**
  - Setup: Bloom Filter of size 128 bits, 20 Number160 objects
- **Applications: Distributed Caching, Spell checking, Routing, (distributed) Databases**
- **B-Tracker uses Bloom Filters**
  - “To avoid duplicates send compressed list of known peers”
  - Idea: store peers in Bloom Filter and send it. Other peers only send us peer not in the Bloom Filter
    - Less traffic (request is larger, reply may be smaller)
    - False positive are possible
- **Demo: Bloom Filter for get() in TomP2P**