HS18

DISTRIBUTED SYSTEMS ADVANCED & BLOCKCHAIN

TomP2P Basics

T. Bocek

thomas.bocek@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Distributed Systems - In the *News*

- **21.09.2018: The Good, the Bad and the Ugly Details of One of Bitcoin's Nastiest Bugs Yet**
  - For over a year, Bitcoin core had a severe bug, was fixed in 0.16.3 and 0.17.0rc4
  - Upgrade if you are running nodes (CVE-2018-17144)
  - Denial of Service inflation vulnerability
    - Bitcoin Core 0.14.X – double spend within the same transaction will crash
    - Bitcoin Core 0.15 – under some circumstances, double spending possible → more than 21mio
  - Timeline:
    - Reported: September 17, 2018, 14:57
    - Slushpool patched system: 20:48
- **Other views**

- **19.09.2018: EU's Copyright Directive and the P2P Internet**
  - Opinion!
  - proactively filter uploaded content "meme ban"
  - "The new Copyright Directive is probably going to harm the internet as we know it, a lot, specially for europeans. **But** it might end up accidentally helping new peer-to-peer web protocols florish. It might end up making the current internet status quo unbearable, creating a gap that can only be filled by modern peer-to-peer protocols."

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
- **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)
  - 11 contributors on github, I'm the main one ☺
  - Don't buy the book!

Introduction

TomP2P

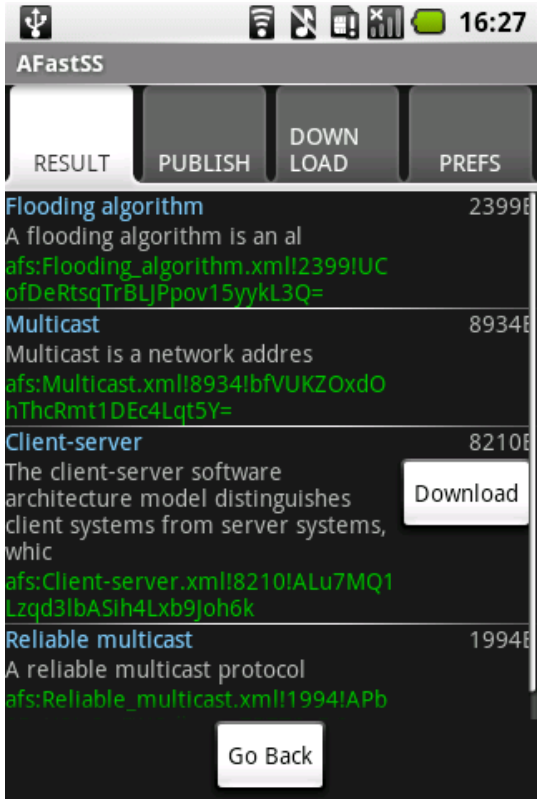
A P2P-based high performance key-value pair storage library

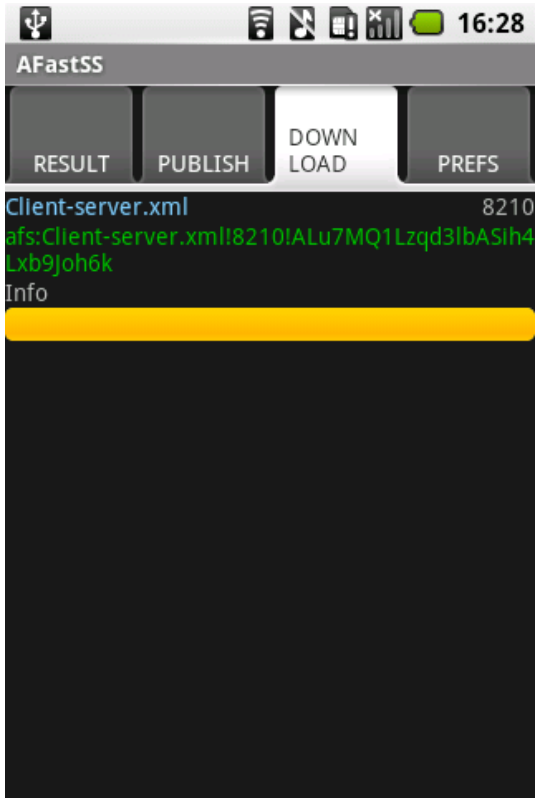
- **Academic background (CSG - UZH):**
  - Used in EU projects: EC-GIN, Emanics, SmoothIT, SmartenIT, Flamingo
  - Used in research projects: LiveShift, DRFS, Radiommender, Box2Box, Hive2Hive, B-Tracker, PiCsMu, peerwasp, (and non-academic)
- **http://tomp2p.net**
  - Specific questions: thomas.bocek-at-hsr.ch or tom -at- tomp2p.net
  - github issues, not in a good state… examples!
  - 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





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

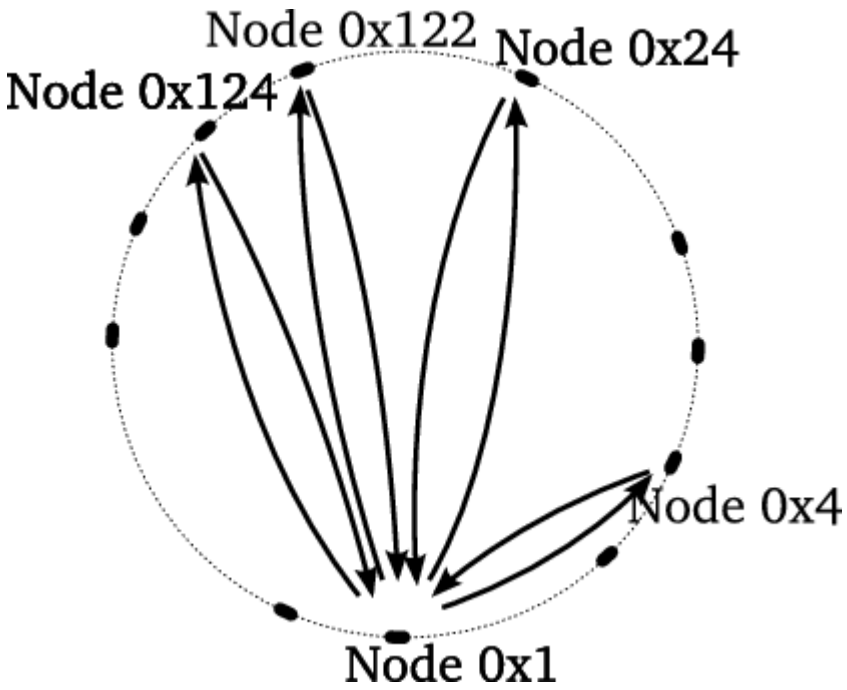
- **Demo: a simple put / get example (example)**
- **Package net.tomp2p.examples. ExamplePutGet**

```
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



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)**
  - New utilities necessary (loops as recursions)
  - 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");
}
```

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**

- **Fun with futures in Java: 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);
			}
		});
}
```

- **.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`

- **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)`

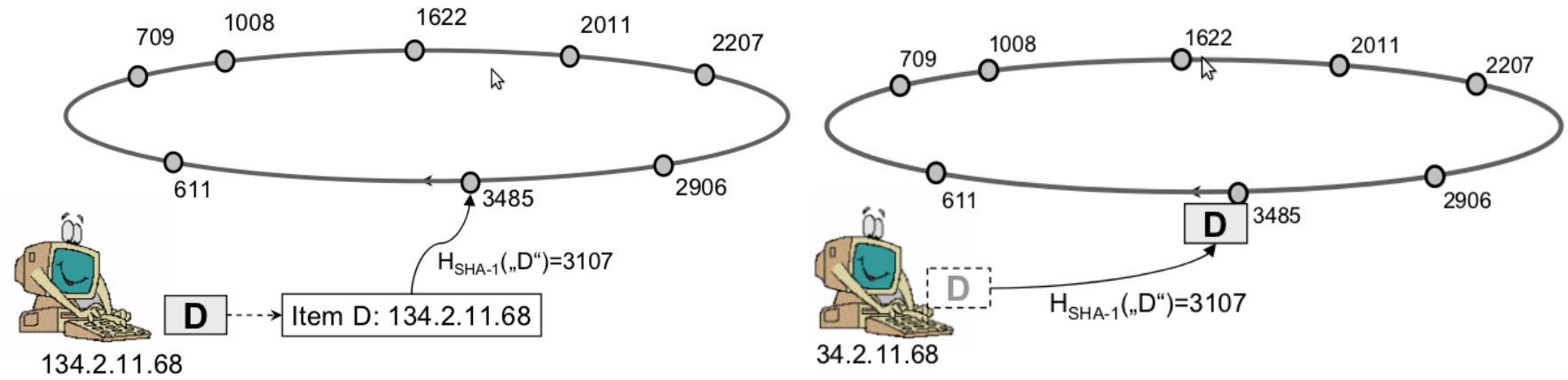
URL: b.socrative.com/student/

Room: HSR

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());
}
```

Questions?

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