Lecture 6

P2P with TomP2P

http://tomp2p.net/doc

Introduction into P2P

*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**
   1. NAT (UPNP/NAT-PMP)
   2. Security
   3. Replication
   4. SimGrid
   5. Direct data / persistent connections
   6. Android
6. **References**

1. Introduction

What is TomP2P

History and project information

Introduction

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

- **TomP2P is an *extended* DHT**
  - Distributed hash table concept → `put(key,value) / get(key)`
  - Extended DHT operations → `put(key1,key2,value) / add(key, value)`
- **TomP2P features (v.4.1)**
  - Java6 DHT implementation with non-blocking IO
  - Replication (direct / indirect)
  - Mesh-based distributed tracker
  - Stores multiple values for one key (examples follow)
  - Storage is memory-based or disk-based

Introduction

- **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
    - Current release (preview) 4.1
    - Latest feature (work in progress) MapReduce
    - 22K LoC, 8K LoC JUnits

Introduction

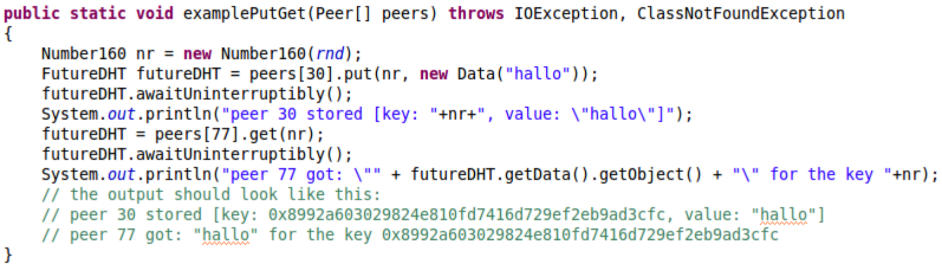
- **Academic background (CSG - UZH):**
  - Used in EU projects: EC-GIN, EMANICS, SmoothIT
  - Used in research projects: FastSS, LiveShift, PSH, B-Tracker, DRFS
- **http://tomp2p.net**
  - For questions: mailinglist (http://lists.tomp2p.net/cgi-bin/mailman/listinfo)
  - Specific questions: bocek -at- ifi.uzh.ch or tom -at- tomp2p.net
  - Documentation: http://tomp2p.net/doc/ (TomP2P v4.0)
    Overview: http://en.wikipedia.org/wiki/TomP2P
    - If something is missing, ask!
  - Development: https://github.com/tomp2p
    - Feature request possible if good reasons provided
- **Demo: how to setup TomP2P with Eclipse/git/maven**

2. Example

Example and Demo

Example

- Demo: a simple put / get example
- Package net.tomp2p.examples.Examples



```java
public static void examplePutGet(Peer[] peers) throws IOException, ClassNotFoundException
{
    Number160 nr = new Number160(rnd);
    FutureDHT futureDHT = peers[30].put(nr, new Data("hallo"));
    futureDHT.awaitUninterruptibly();
    System.out.println("peer 30 stored [key: "+nr+", value: \"hallo\"]");
    futureDHT = peers[77].get(nr);
    futureDHT.awaitUninterruptibly();
    System.out.println("peer 77 got: \"" + futureDHT.getData().getObject() + "\" for the key "+nr);
    // the output should look like this:
    // peer 30 stored [key: 0x8992a603029824e810fd7416d729ef2eb9ad3cfc, value: "hallo"]
    // peer 77 got: "hallo" for the key 0x8992a603029824e810fd7416d729ef2eb9ad3cfc
}
```

3. Fundamental Concepts

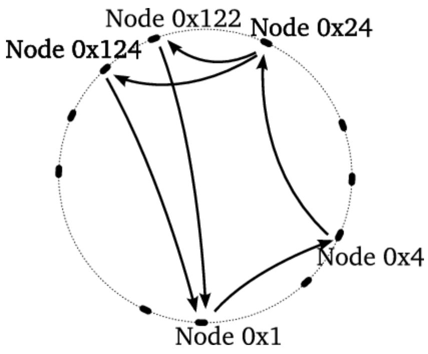
XOR-based iterative routing

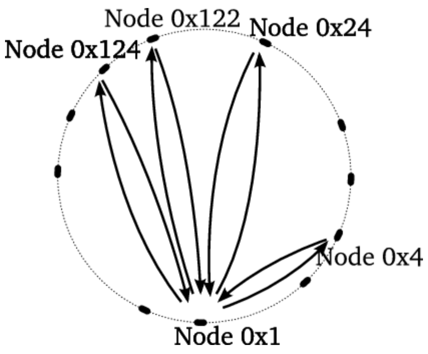
Futures

API Overview

Fundamental Concepts

- Recursive routing vs. iterative routing





+ online status update

- faulty peers cause delay

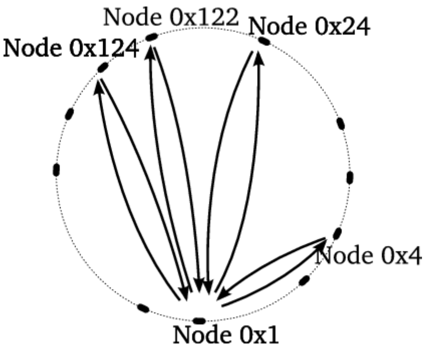
+ control

- neighbor maintenance

Fundamental Concepts

- **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



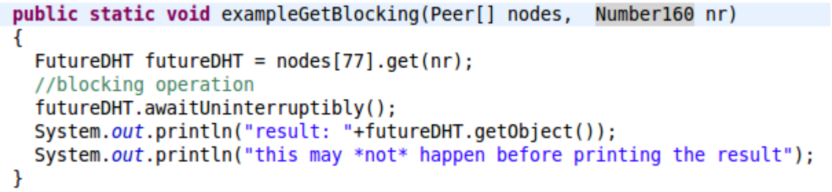
    - Difference to Pastry: one metric, no leaf set / routing table

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 12
  - By default UDP, message header 56 bytes
  - Configuration options (RoutingConfiguration.java)
    - directHits – used for get() operations. (routing sends digest)
    - forceTCP – use TCP instead of UDP
    - maxSuccess, maxFailure – stop conditions
    - parallel – number of parallel connections
    - maxNoNewInfoDiff – stop condition. Stops if no new information was reported. Difference to minimumResults (e.g. for `get(key)`)
  - For the CT - don’t worry, default settings are fine ☺

Fundamental Concepts

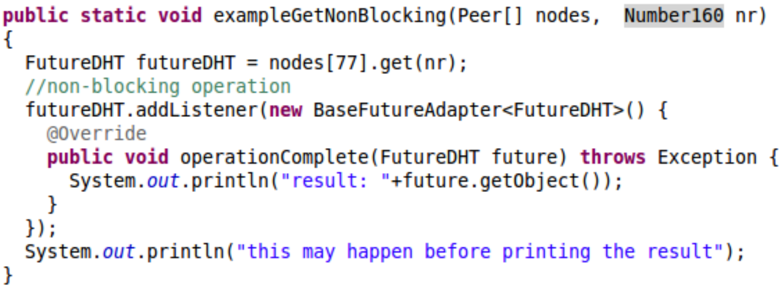
- **All distributed operations use futures**
- **Future objects**
  - Keeps track of future events, while the “normal” program flow continues → `addListener()` or `await()`
  - `await()`: Operations are executed in same thread
  - `addListener()`: Operations are executed in same or other thread
- **Demo: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



```java
public static void exampleGetBlocking(Peer[] nodes,  Number160 nr)
{
  FutureDHT futureDHT = nodes[77].get(nr);
  //blocking operation
  futureDHT.awaitUninterruptibly();
  System.out.println("result: "+futureDHT.getObject());
  System.out.println("this may *not* happen before printing the result");
}
```

Fundamental Concepts

- **Demo: non - blocking operation (net.tomp2p.examples.Examples)**
  - New utilities necessary (loops as recursions)
  - Advise: use `addListener(…)` as much as possible!
  - `operationComplete(…)` must be **always** called



```java
public static void exampleGetNonBlocking(Peer[] nodes,  Number160 nr)
{
  FutureDHT futureDHT = nodes[77].get(nr);
  //non-blocking operation
  futureDHT.addListener(new BaseFutureAdapter<FutureDHT>() {
    @Override
    public void operationComplete(FutureDHT future) throws Exception {
      System.out.println("result: "+future.getObject());
    }
  });
  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
  - `FutureWrapper()`
    - A placeholder for futures that are created later
- **ForkJoin in Java7**
  - Fork and join framework – future utilities in TomP2P focus on join, forking is done “manually”

Fundamental Concepts

- **Fun 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

- **API Overview: Peer.java**
  - Basic methods for DHTs
    - `put(key, value)`,`get(key)`
  - Additional methods in TomP2P:
    - For initial connection: `boostrapBroadcast()` / `boostrap(Ipaddress, port)` / `discover(IPaddress, port, port)`
    - Requires to specify `set*DataReply(…)`: `send(peeraddress, value)` / `send(peerconnection, value)` / `send(key, value)`
    - Data manipulation: `add(key, value)` / `putIfAbsent(key, value)` / `digest(key)` / `remove(key)`
    - Tracker operations: `getFromTracker(key)` / `addToTracker(key, value)`
    - Used mostly internally `parallelRequests(…)`

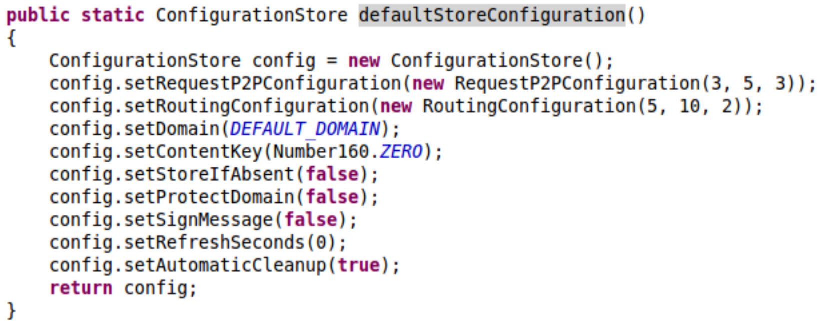
Fundamental Concepts

- **Configurations used in the API**
  - TomP2P can store multiple values for a key
    - `put(location_key, content_key, value)` → content_key specified in Configurations
    - `get(location_key)`
      → 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 Configurations
    - `put(key, domain, value)` → `get(key,domain)`

Fundamental Concepts

- **Configurations Example**

```
Number160 nr = new Number160(rnd);
ConfigurationStore cs = Configurations.defaultStoreConfiguration();
cs.setDomain(Number160.createHash("my_domain"));
cs.setContentKey(new Number160(11));
FutureDHT futureDHT = peers[30].put(nr, new Data("hallo"), cs);
```



```
public static ConfigurationStore defaultStoreConfiguration()
{
    ConfigurationStore config = new ConfigurationStore();
    config.setRequestP2PConfiguration(new RequestP2PConfiguration(3, 5, 3));
    config.setRoutingConfiguration(new RoutingConfiguration(5, 10, 2));
    config.setDomain(DEFAULT_DOMAIN);
    config.setContentKey(Number160.ZERO);
    config.setStoreIfAbsent(false);
    config.setProtectDomain(false);
    config.setSignMessage(false);
    config.setRefreshSeconds(0);
    config.setAutomaticCleanup(true);
    return config;
}
```

5. Components with Examples

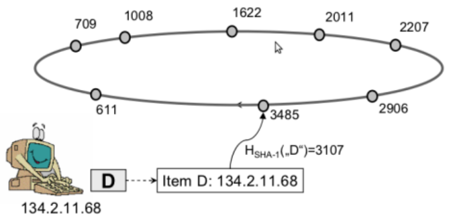
DHT

Tracker

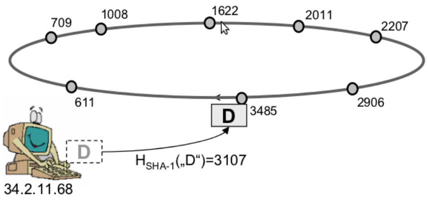
Components with Examples

- **DHT vs. Tracker**
  - M03, slide 23: DHT “stored by value” – direct storage
  - M03, slide 24: Tracker “stored by reference” – indirect storage

indirect (Tracker)



direct (DHT)

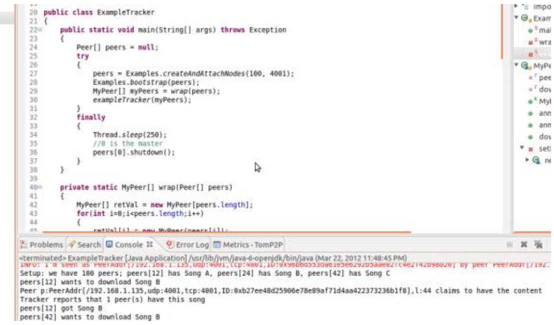


Components with Examples

- **B-Tracker**
  - Centralized tracker – one peer 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
  - Currently tests with B-Tracker in Vuze

Components with Examples

- **Demo: Tracker with exchange of popular items (net.tomp2p.examples.ExampleTracker)**
  - Creat 100 peers, 3 peers have initially each a song
  - M03 slide 26: peer joining / bootstrap



```java
public class ExampleTracker
{
    public static void main(String[] args) throws Exception
    {
        Peer[] peers = null;
        try
        {
            peers = Examples.createAndAttachNodes(100, 4001);
            Examples.bootstrap(peers);
            MyPeer[] myPeers = wrap(peers);
            exampleTracker(myPeers);
        }
        finally
        {
            Thread.sleep(250);
            //0 is the master
            peers[0].shutdown();
        }
    }

    private static MyPeer[] wrap(Peer[] peers)
    {
        MyPeer[] retVal = new MyPeer[peers.length];
        for(int i=0;i<peers.length;i++)
        {
            retVal[i] = new MyPeer(peers[i]);
```

```
<terminated> ExampleTracker [Java Application] /usr/lib/jvm/java-6-openjdk/bin/java (Mar 22, 2012 11:48:45 PM)
INFO: I'm seen as PeerAddr[/192.168.1.135,udp:4001,tcp:4001,ID:[illegible]] by peer PeerAddr[/192.
Setup: we have 100 peers; peers[12] has Song A, peers[24] has Song B, peers[42] has Song C
peers[12] wants to download Song B
Peer p:PeerAddr[/192.168.1.135,udp:4001,tcp:4001,ID:0xb27ee48d25906e78e89af71d4aa422373236b1f8],l:44 claims to have the content
Tracker reports that 1 peer(s) have this song
peers[12] got Song B
peers[42] wants to download Song B
```

Components with Examples

- **Demo: Tracker with exchange of popular items**
  - Although demo uses `await()`, try not to use it
- **Demo: Store popular items in DHT (net.tomp2p.examples.ExampleDHT)**
  - Tracker vs. DHT what is better for the CT? You decide!
- **Further interesting aspects for the challenge task:**
  - Automate downloads
  - Suggestions evaluated by the user
  - How to do this more anonymous: music list from a peer is known
  - Incentives
  - Spamming the system with bogus suggestions

6. Advanced Topics

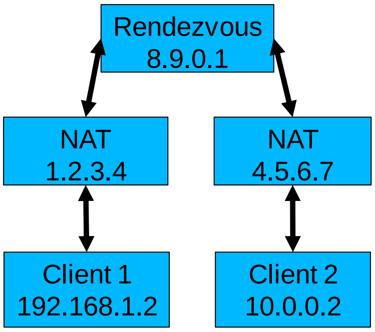
NAT (UPNP/NAT-PMP)
Security
Replication
SimGrid integration
Direct data connection / persistent connection
Android

Advanced Topics

- **NAT**
  - Network Address Translation – breaks end-to-end
  - “If nothing else, [NAT] can serve to provide temporarily relief while other, more complex and far-reaching solutions are worked out” (RFC 1631 - The IP Network Address Translator (NAT))
  - Easy solutions: UPNP / NAT-PMP
    - Both configure port forwarding, but UPNP is more
    - UPNP: discover devices - uses broadcasting to find router (Simple Service Discovery Protocol)
    - UPNP: configure devices - uses HTTP and XML to configure portforwarding (Internet Gateway Device Protocol)
    - NAT-PMP: protocol made for configuring port-forwarding, but no discover (how to find router?)

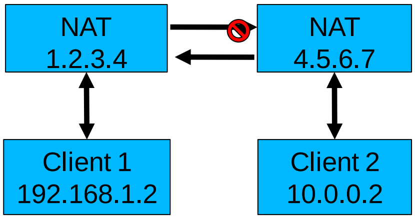
Advanced Topics

- **NAT: Difficult solution: rendezvous / relay peer which does “hole punching”, in worst case relay traffic.**
- Hole punching
  - ▶ Client 1 wants to connect to Client 2 (both clients maintain connection to Rendezvous)
  - ▶ Client 1 sends connection request to Rendezvous → Redezvous send connection request to Client 2 and the outgoing port X that Client 1 will use and send to Client 1 what outigoing port Y will be used by Client 2 (guess!)



Advanced Topics

- Hole punching
  - ▶ Client 1 sends request to NAT 4.5.6.7 that will fail – no mapping, however, Client 1 creates a mapping with that request
  - ▶ Client 2 send a request to Client 1 (1.2.3.4:X) – success!



Mapping for NAT 1.2.3.4 (Client 1)		
192.168.1.2:4000	1.2.3.4:X	4.5.6.7:Y

Mapping for NAT 4.5.6.7 (Client 2)		
10.0.0.2:5000	4.5.6.7:Y	1.2.3.4:X

Advanced Topics

- **NAT example in TomP2P, the easy solution**
  - TomP2P supports NAT-PMP and UPNP, no holepunching or relaying
  - Before bootstrap: `peer.discover(PeerAddress);`
  - How it works: (1) send request how others peers sees our IP
    - If other peers sees the same IP as we see, we are fine
    - If not, we are most likely behind a NAT
  - (2) do UPNP, if it fails, do NAT-PMP, if it fails, no connection
  - (3) If it works test connection, send request to other peer to contact us using the port we just set up.
  - (4) If we get contacted by this peer within 5 sec, port-forwarding works.
  - Manual setup possible using `Bindings.java`
- **No Demo, did not bring my NAT device**

Advanced Topics

- **Security in TomP2P**
  - Signature-based, no data encryption
  - Messages are signed using SHA1 with DSA
  - Sybil attacks!
    - This attack creates large number of identities, may collude
- **How to prevent Data from being overwritten**
  - Domain and entry protection, requires cooperation
  - `StorageGeneric.setProtection(…)`

For domains and entries		
protectionEnabled	ALL	NONE
protectionMode	NO_MASTER	MASTER_PUBLIC_KEY

Advanced Topics

- **Domain protection**
  - Set publick key `new PeerMaker(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)
  - Same concept for entries
  - Tracker should have no domain protection and entry protection set to Enable=NONE, Mode=MASTER_PUBLIC_KEY → WiP
- **Demo**

Advanced Topics

- **Demo 1 (net.tomp2p.examples.ExampleSecurity):**
  - 3 peers, all with public keys
  - Setup for domains: Enable=ALL, Mode=MASTER_PUBLIC_KEY
  - (1) peer1 stores data in domain2 → success
  - (2) peer3 wants to store data in domain2 → fail
  - (3) peer2 wants to store data in domain2 → success
- **Demo 2 (net.tomp2p.examples.ExampleSecurity):**
  - 3 peers, all with public keys
  - Setup for domains: Enable=NONE, Mode=MASTER_PUBLIC_KEY
  - (1) peer1 stores data in domain2 → success
  - (2) peer3 wants to store data in domain2 → success
  - (3) peer2 wants to store data in domain2 → success
  - (4) peer3 wants to store data in domain2 → fail

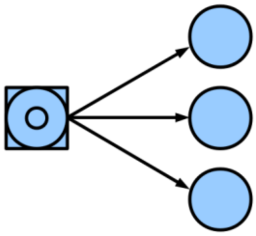
Advanced Topics

- **Replication**
  - Enough replicas
  - Direct replication
    - Originator peer is responsible
    - Periodically refresh replicas
    - Example: tracker that announces its data

□ Responsible for X

◉ Originator of X

○ Close peers to X



- **Problem**
  - Originator offline → replicas disappear. Content has TTL, e.g. `data.setTTLSeconds(15)`

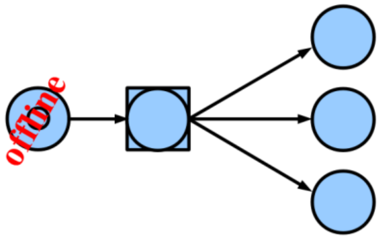
Advanced Topics

- **Indirect Replication**
  - The closest peer is responsible, originator may go offline
    - Periodically checks if enough replicas exist
    - Detects if responsibility changes

□ Responsible for X

◉ Originator of X

○ Close peers to X



- **Problem**
  - Requires cooperation between responsible peer and originator
  - Multiple peers may think they are responsible for different versions → eventually solved

Advanced Topics

- **Replication Demo (net.tomp2p.examples.ExampleDirectReplication)**
  - Direct replication – for `put()` and `add()`
    - `ConfigurationStore.setRefreshSeconds(2);`
    - Stop replication if in progress: `futureDHT.shutdown();`
  - Direct replication for `remove()`
    - `ConfigurationRemove.setRefreshSeconds(2);`
    - `ConfigurationRemove.setRepetitions(2);`
    - Stop replication after 4 seconds, 2 repetitions
  - Indirect replication (net.tomp2p.examples.ExampleIndirectReplication)
    - Set when creating peers
    - `PeerMaker.setEnableIndirectReplication(true);`
    - Two types of events: (1) peer learns about closer peer (2) peer checks frequently for enough replicas

Advanced Topics

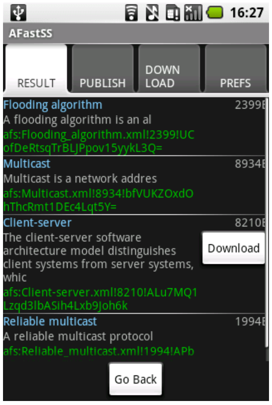
- **SimGrid integration**
  - Scalable simulation of distributed systems
  - Publish over 100 papers that include SimGrid
  - SimGrid vs. real network
  - For TomP2P: simulates network with many peers
    - Defined in XML files: `platform.xml` and `deployment.xml`
  - Logging in console
  - Current issue in jMSG: threads, threads, threads!
- **Demo: how to use it with TomP2P**
  - Get the Eclipse workspace: http://tomp2p.net/dev/simgrid/ (Linux x64)
  - 10’000 peers are OK, to simulate more, kernel parameter tuning

Advanced Topics

- **Direct data and persistent connections**
  - All connections in TomP2P are RPC and very short-lived
    - Open connection – request – reply – close connection
  - Direct data as seen in the tracker example → keep alive
  - Direct `send(PeerAddress, …)` or with routing `send(key, …);`
  - Always use `setObjectDataReply()` or `setRawDataReply()`
    - Object serializes object to byte[] (easy)
    - Raw exposes (Netty) buffer to the user for your own protocol (more work)
  - Persistent connections set by the user
    - Only for direct send `send(PeerAddress, …)`
- **Demo with persistent connections (net.tomp2p.examples.ExamplePersistentConnection)**

Advanced Topics

- **TomP2P with Android (early research)**
  - CSG: early adopter



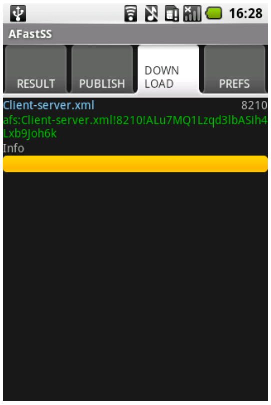
16:27
AFastSS
RESULT | PUBLISH | DOWN LOAD | PREFS

Flooding algorithm 2399E
A flooding algorithm is an al
afs:Flooding_algorithm.xml!2399!UCofDeRtsqTrBLJPpov15yykL3Q=

Multicast 8934E
Multicast is a network addres
afs:Multicast.xml!8934!bfVUKZOxdOhThcRmt1DEc4Lqt5Y=

Client-server 8210E
The client-server software architecture model distinguishes client systems from server systems, whic
afs:Client-server.xml!8210!ALu7MQ1Lzqd3lbASih4Lxb9Joh6k

Reliable multicast 1994E
A reliable multicast protocol
afs:Reliable_multicast.xml!1994!APb



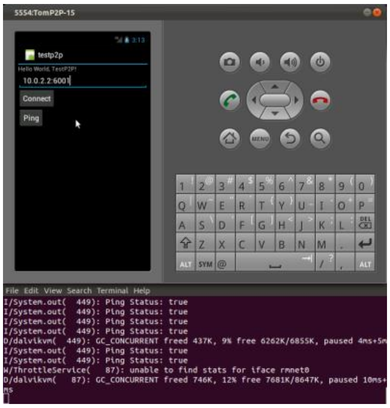
16:28
AFastSS
RESULT | PUBLISH | DOWN LOAD | PREFS

Client-server.xml 8210
afs:Client-server.xml!8210!ALu7MQ1Lzqd3lbASih4Lxb9Joh6k
Info

Advanced Topics

- **TomP2P with Android ICS 4.0.3**
  - Latest Android is ~Java6 (source code) compatible
  - Extra work (permissions, IPv4)
  - TomP2P with multiple emulators
    - Redirect ports!
    - telnet to all emulators:

      ```
      redir add udp:x:y
      redir add tcp:x:y
      ```
    - Connect to 10.0.2.2!
- **TomP2P on Android: Demo with local peers**



```
I/System.out(  449): Ping Status: true
I/System.out(  449): Ping Status: true
I/System.out(  449): Ping Status: true
I/System.out(  449): Ping Status: true
D/dalvikvm(  449): GC_CONCURRENT freed 437K, 9% free 6262K/6855K, paused 4ms+5m
I/System.out(  449): Ping Status: true
I/System.out(  449): Ping Status: true
W/ThrottleService(   87): unable to find stats for iface rmnet0
D/dalvikvm(   87): GC_CONCURRENT freed 746K, 12% free 7681K/8647K, paused 10ms+
ms
```

7. References

4. References

- **TomP2P homepage**
  - http://tomp2p.net
- **Kademlia**
  - S. C. Rhea and J. Kubiatowicz: Probabilistic Location and Routing; IEEE INFOCOM 2002, New York, NY, USA, pp. 1248-1257, June 2002.
- **Sybil attack**
  - Douceur, John R. (2002). "The Sybil Attack". International workshop on Peer-To-Peer Systems. Retrieved 31 March 2011.
- **Sybil attack counter measurements**
  - Raul Landa , David Griffin , Richard G. Clegg , Eleni Mykoniati , Miguel Rio. "A Sybilproof Indirect Reciprocity Mechanism for Peer-to-Peer Networks"
- **B-Tracker**
  - Hecht, F.V.; Bocek, T.; Stiller, B.; B-Tracker: Improving load balancing and efficiency in distributed P2P trackers, Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on , vol., no., pp.310-313, Aug. 31 2011-Sept. 2 2011
- **Fork/Join in Java**
  - http://docs.oracle.com/javase/tutorial/essential/concurrency/forkjoin.html
- **Hole punching**
  - http://www.brynosaurus.com/pub/net/p2pnat/