Getting started Using ANts to build up p2p software

Introduction

This document wants to be just a simple introduction to ANts as a component library for software development.
Even if you can run the jar and use the program through its IDE ANts has been concepted as a structured library that cover many aspects of p2p.
I want to introduce the reader to the outlines of this component and its base uses.

The class Ant

In the package ants.p2p you can find some classes (the "public ones"), through these classes you can access to almost every feature of ANts. Let's have a closer look to Ant.class.
This class realizes a peer low level structure. What do I mean with low level?
An Ant instance gives only the basic communication instruments:

1) Data messages
2) Ack messages
3) Broadcast messages

It also encapsulates everything related to low level security (peer to peer cryptography).
And gives the user the methods needed to set up a peers net:
addNeighbour(...)

When you connect an Ant instance to a neighbour a DH-KA is performed and a secure channel is set up between the two peers. So you can use this channel to send messages to the other peers routing through the peer you are connected to using the method sendMessage(...).
The messages you can send this way are instances of the base class Message. These messages are not crypted and contain only routing infos (source, dest and infos about timeToLive etc.). You can also send broadcast messages using the method sendBroadcaseMessage(...), but we are not talking about this by now. We are not talking also about the routing procedure... I'll write a separate document about that. Let's assume you have a routing system that lets you reach your dest node.

I said you can send simple messages. A message, once sent, tries to reach the destination, if it does reach it an ack message is sent back through the SHORTEST route. Only if this ack reaches the source within a determined interval (timeout) the message is considered routed.
So we have two constraints: a timeout on the source node and a timeToLive on the message (if the message timeToLive, i.e. numbers of hops thorugh peers, expires the message is discarded). A particular consistency policy is applied to grant an "at most once" semantic in message sending. So if you send a message, your ant instance can also perform retransmissions and this will not cause an inconsistent state on the destination node due to message duplicates.

So an Ant instance gives you the base behaviour on which you can build a more structured and secure net.

So you can create some Ant instances and connect them by the addNeighbour() method.

```
Ant n1 = new Ant("1", 10, 4001);
Ant n2 = new Ant("2", 10, 4002);

n1.addNeighbour("localhost",4002,false);
```

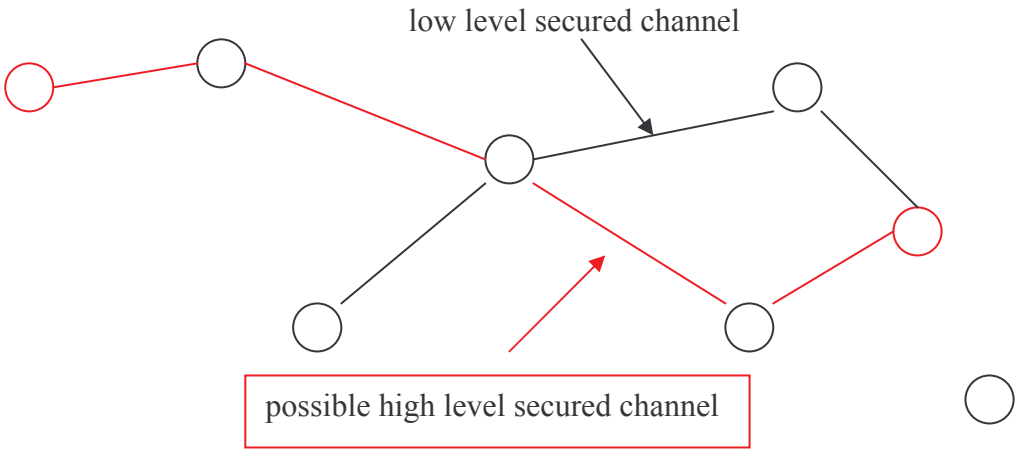
then you can send messages through the sendMessage(...) method.

```
n1.sendMessage(new Message(), "2", false);
```

this just send an empty message to the dest... not so useful :)
Let's consider a more structured Ant...

The class WarriorAnt

This class extends the base Ant and adds features related to endpoint security and complex data management.
You can do the same things you have already done with the Ant class... but now you can also set up secure channels with other peers distributed on the net by performing an high level DH-KE.



I write "possible" because the route is not a deterministic one... the security lies on the endpoints.

So you can instantiate two WarriorAnt:

```
WarriorAnt n1 = new WarriorAnt("1", 10, 4001);
WarriorAnt n2 = new WarriorAnt("2", 10, 4002);

n1.addNeighbour("localhost",4002,false);
```

and set up a connection...

you can also instantiate a third ant:

```
WarriorAnt n3 = new WarriorAnt("3", 10, 4003);
```

And connect it to the second:

```
n2.addNeighbour("localhost",4003,false);
```

then you can set up a secure cannel between the first and the third through the second:

```
n1.createSecureConnection("3");
```

Now you have a secure high level channel.
Through this channel you can pull & push files... (note that IDs are not linked to IPs, this grants you cannot be tracked... see the MUTE specifications for more particulars).

But what about queries? Queries are performed using the sendBroadcast() feature... so how can we grant privacy?
The QueryMessage extends the base one... it contains a dinamically generated public key of the user performing the query. The broadcasted query is processed in a sequential way passing through a determined number of nodes (in accord to the accuracy decided by the user). Each node that process the QueryMessage generates a sessionKey and crypts the query result inserted in the query message using that key... then encrypts the session key with the public key of the query issuer. At last the query message is returned to the issuer.

These are basically the main features of the system... there's still a lot to say about it (settings, multiple download consistency policy, structured query support with SQL like operators etc.).

I will write separate documents about every single feature... by now I just insert a code example that shows the code to set up of a small net and the client code that performs operations on the net itself.

```
Class to set up a net with 6 peers

package ants.test;
import ants.p2p.*;
import java.io.*;
import ants.p2p.utils.*;

public class NetAntsTest {

  public static void main(String[] args) {
    try{
      //This runs the indexer for the directory specified
      //only file indexed can be pulled down so this grants
      //security over random pulls performed:
      //i.e. a pull like "../../mySecretFile.txt"
      BackgroundEngine be = BackgroundEngine.getInstance();
      be.addDirectory(new File("c:\\share\\musica\\canzoni"));

      //Same things we've seen above
      WarriorAnt n1 = new WarriorAnt("1", 10, 4001);
      WarriorAnt n2 = new WarriorAnt("2", 10, 4002);
      WarriorAnt n3 = new WarriorAnt("3", 10, 4003);
      WarriorAnt n4 = new WarriorAnt("4", 10, 4004);
      WarriorAnt n5 = new WarriorAnt("5", 10, 4005);
      WarriorAnt n6 = new WarriorAnt("8", 10, 4006);
```

```
       Thread.sleep(1000); //Give the Ant the time :P
       n1.addNeighbour("localhost",4002,false);
       Thread.sleep(1000);
       n2.addNeighbour("localhost",4003,false);
       Thread.sleep(1000);
       n3.addNeighbour("localhost",4004,false);
       Thread.sleep(1000);
       n4.addNeighbour("localhost",4005,false);
       Thread.sleep(1000);
       n5.addNeighbour("localhost",4006,false);

      //This let you print the state of the n3 ant
       BufferedReader bis = new BufferedReader(new InputStreamReader(System.in));
             String to;
             while ( (to = bis.readLine()) != null) {
               try {
                 if (to.equals("state")) {
                   System.out.println("Pending pull requests: " +
                                      n3.pendingFilePullRequests.size());
                   System.out.println("In service pull requests: " +
                                      n3.inServiceFilePullRequests.size());
                   System.out.println("In service push requests: " +
                                      n3.inServiceFilePushRequests.size());
                 }
               }
               catch (Exception ex) {
                 ex.printStackTrace();
               }

             }
           }
           catch (Exception e) {
             e.printStackTrace();
           }
         }
       }
```

Class to perform operations over the created net.

```
       package ants.test;
       import ants.p2p.*;
       import java.io.*;
       import java.beans.*;
       import ants.p2p.messages.*;
       import ants.p2p.query.*;
       import ants.p2p.query.security.*;
       import ants.p2p.utils.*;

       public class CommandLineAnt {
         public static void main(String[] args) {
           try{
             if(args.length%2!=0)
               return;
             Thread.sleep(1000);
             //first arument is the ID, second is the port
             WarriorAnt n = new WarriorAnt(args[0], 5,
       Integer.parseInt(args[1]));
             //Cycle to add neighbours as specified in the next arguments
             //args[x] = address  args[x+1] = port
```

```
      for(int x = 2; x < args.length; x=x+2){
        n.addNeighbour(args[x], Integer.parseInt(args[x+1]),true);
      }

      //Cycle to read and execute operations
      BufferedReader bis = new BufferedReader(new
InputStreamReader(System.in));
      String to;
      while((to=bis.readLine())!=null){
        try {
          if (to.equals("secure")){// Create a secure connection to a dest
            to = bis.readLine();
            n.createSecureConnection(to);
          }
          else if (to.equals("state")) {// Print the state

            System.out.println("Pending pull requests: " +
                               n.pendingFilePullRequests.size());
            System.out.println("In service pull requests: " +
                               n.inServiceFilePullRequests.size());
            System.out.println("In service push requests: " +
                               n.inServiceFilePushRequests.size());
          }
          else if (to.equals("size")) {// Get the size of an indexed file
            to = bis.readLine();
            String ha = bis.readLine();
            n.pullFileSize(to, ha, "8");
          }
          else if (to.equals("query")) {// Do a query
            to = bis.readLine();
            String ttl = bis.readLine();
            QueryStringItem qsi = new QueryStringItem(null, to);
            AsymmetricProvider ap = new AsymmetricProvider(true);
            ap.storeMessageHeader();
            n.doQuery(qsi, ap.getPublicHeader(),Long.parseLong(ttl));
          }
          else if (to.equals("broad")) {// Broadcast an empty message
            Message m = new Message();
            MessageWrapper wm = n.sendBroadcastMessage(m);
          }
          else if (to.equals("multiple")) {// Multiple download
            to = bis.readLine();
            String ha = bis.readLine();
            //Have a look at the code to understand the formal parameters
            MultipleSourcesDownloadManager msdm = new
MultipleSourcesDownloadManager(n,to,ha,0L,(int)Math.pow(2, 18),4788671);
            msdm.addPeer("4");
            msdm.addPeer("3");
            msdm.addPeer("2");
            msdm.addPeer("1");
            msdm.start();
          }
          else if (to.equals("pull")) {// Single download
            String file = bis.readLine();
            String hash = bis.readLine();
            String from = bis.readLine();
            n.pullFile(file, hash, 0, Long.MAX_VALUE, from,
                       (int) Math.pow(2, 19), "00.tmp", false);
          }
        }
        catch (Exception ex) {
          ex.printStackTrace();
        }
```

```
        }
      }catch(Exception e){e.printStackTrace();}
    }
  }
```