Exercise 4

Exercises Peer-to-Peer-Systems and Security (SS2011)

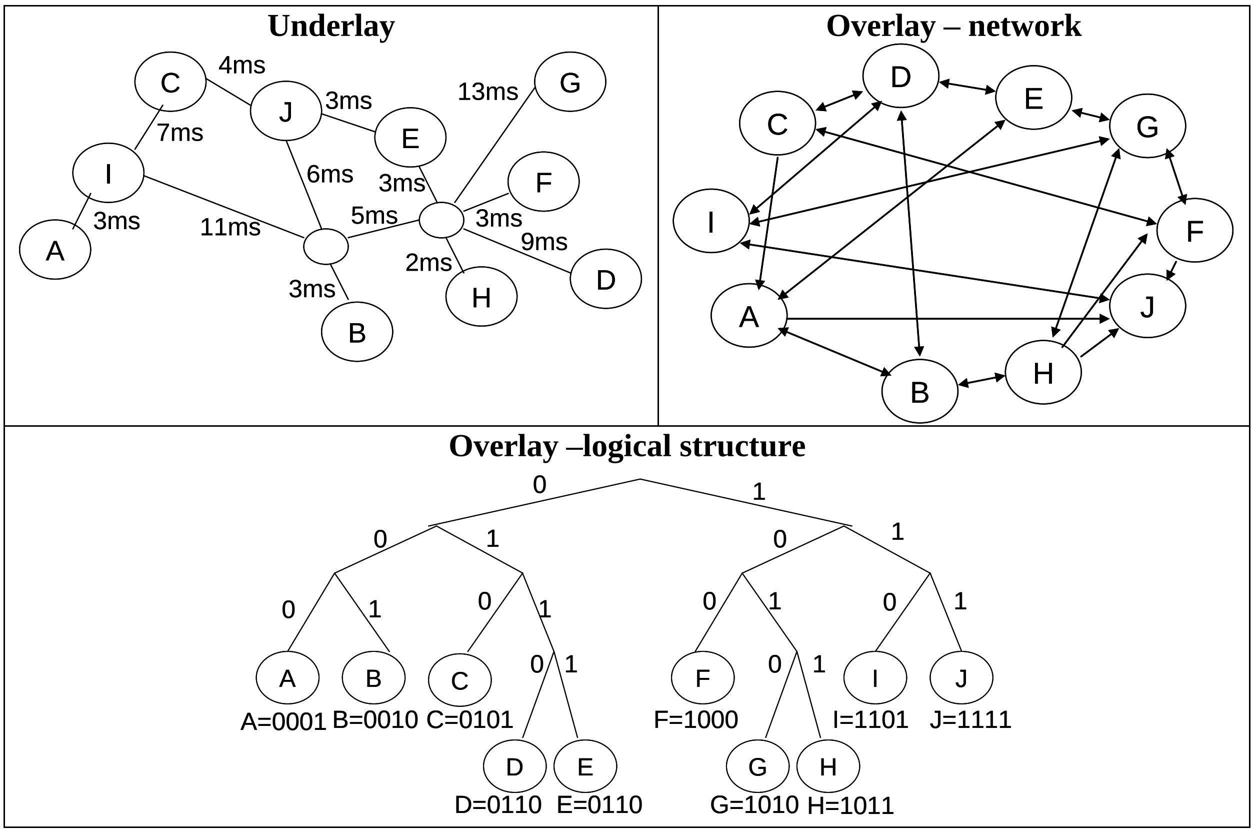
Dr. Heiko Niedermayer
Lehrstuhl für Netzarchitekturen und Netzdienste
Technische Universität München

Monday 20.6 2011
Hand-in: Monday 4.7. 2010 in lecture or per mail
Exercise: 7.7. 2010

<u>Rules:</u> There will be five exercise sheets. You have to hand-in 70 % of the assignments, attend atleast 3 exercise courses and present a solution in the exercise course to get the 0.3 bonus.

Task 1 Stretch and Proximity Neighbor Selection

The Figure below shows the underlay including the latencies along different paths. The second image is the structure of the overlay (nodes that link to each other) and the image on the bottom is the logical structure if we assume that our network uses a routing table like in Pastry, but without neighbor and leaf set.
In this task, stretch always refers to latency. Always state what path is used when you calculate the stretch.



a) What is the stretch (with respect to latency) for queries from F to A and from B to E?
b) Assume that each node on the way now applies PNS for its routing table. This means that it selected the latency-optimal node in each subtree. What is now the stretch for the queries from a)?

Solution:

a)
F->A:
- direct: 22 ms

- via overlay (via C): 13 + 10 ms
Stretch_FA = 23/22 = 1,045

B->E
- direct: 11 ms
- via overlay (via D): 29 ms
Stretch_BE = 29/11 = 2,63

b)
F->A
PNS from F to subtree 0x: to E in 6ms
PNS from E to subtree 00x: to B in 11 ms
B->A: in 17ms
Stretch_FA_PNS = 34 / 22 = 17 / 11 = 1,54 (got worse due to more hops!)

B->E
PNS from B to subtree 01x: E in 11 ms
Stretch_BE_PNS = 1,0

Please note: in task b) the goal was to change the overlay structure according to PNS using the logical structure. So, for F you have to look what is the best node in 0* and then select this node as F's neighbour in the new overlay structure after PNS. Here, we assumed that the best nodes of all nodes in the corresponding intervals are selected. In reality one may check the distance (e.g. RTT) only to a fixed number of nodes and select them the best among them for the link.

Task 2 Authentication

In this task we specify a cryptographic protocol which is meant to be used for mutual authentication.

a) Specify on what information and when in the protocol do the entities A,B, and S detect a successful authentication run?

b) The protocol is insecure. Find an attack. (The strength of the attacker is that it can read, send, fake, and drop messages in the network, yet it cannot break cryptography. This is a common security model in network security.)

Protocol for Task 2:

Prerequisites:

S is a TTP.

Each participant X has a shared key with S. This key is called k kXS.

Let kab = Nb.

Protocol:

```
A -> B: Na, A
B -> S: Na, A, B,{Nb}kBS
S -> A: B, {Na, Nb}kAS
A -> B: {Nb}kab
```

Solution:

A recognizes B: in step 3, the server sends a message which contains Bob and Na

B recognizes A: in step 4 by receiving Nb from A which was protected with the shared key kab

S does not detect the success of a run. It might recognize B on the knowledge of the kBS encrypted and integrity protected). It does not recognize A, but provides information on Nb that only A should be able to read.

Attack:

- The attacker replays an encrypted nonce from a previous interaction of the attacker C_B as normal entity C.

```
1. C -> B: Nc,C
2. B -> S: Nc,C,B,{Nb}kBS
3. S -> C: B,{Nc,Nb}kCS
4. C -> B: {Nb}kcb
Now  Alice(A)  wants  to  communicate  with  Bob(B).
Attacker C_B is sitting in between.
1. A -> C_B: Na,A
2. C_B -> S: Na,A,B,{Nb}kBS
3. S -> A  : B, {Na,Nb}kAS
4. A -> C_B: {Nb}kab
 => A recognized C_B as Bob, C_B also knows kab and
can know communicate with Alice as Bob.
```

Task 3 Authentication

Do the same as in task 2, yet with a different protocol. Hint: Use information from previous runs to attack the protocol. Sig_X stands for encryption with private key.

Prerequisites: S is TTP and for each participant X S knows the corresponding public key PK_X. All participants know the public key PK_S of S.
Let kab = hash(Na,Nb).
Protocol:

```
A -> S: A, Enc_PK_S(Na, B)
S -> A: PK_B, Enc_PK_A(Sig_S(Na, A))
A -> B: Enc_PK_B(Na, A, B, Sig_S(Na, A))
B -> A: Nb, {Na}kab
A -> B: {Nb}kab
```

Solution:

B recognizes A in the 5th message due to the signature from the server in message 3 and the knowledge of nb and kab.
A recognizes B in the 4th message due to the knowledge of Na and kab.
S does not detect success of the protocol run. It does not recognize A, but only provides information that only A can read, its signature of Na and A encrypted with A's public key.

Attack:

- The attacker C_A uses a previous communication of A and C_A to attack A and B.

  A communicates with C (later to become C_A).

  ```
  1. A -> S: A, Enc_PK_S(Na,C)
  2. S -> A: PK_C,Enc_PK_A(Sig_S(Na,A))
  3. A -> C: Enc_PK_C(Na,A,C,Sig_S(Na,A))
  4. C -> A: Nc,{Na}kab
  5. A -> C: {Nc}kab
  ```

  Now C attacks A and B by impersonating A. Thus, we will call C now C_A. It reuses the old nonce Na and signature from the server that it received in the 3rd message. C_A knows the public key PK_B of B (if not, it could ask S and receives it in protocol step 2).

  ```
  3. C_A -> B: Enc_PK_B(Na,A,B,Sig_S(Na,A))
  4. B -> C_A: Nb,{Na}kab
  5. C_A -> B: {Nb}kab
  ```

  => B accepts C_A as A and C_A knows kab.

Task 4 Some questions
Answer the questions with knowledge from the lecture.

a) Cryptographic identities seem to make authentication a lot easier. Let assume, we use cryptographic identities. Do we still need a Certificate Authority? If yes, for what? If no, why not?
b) Why is trust important for key distribution?
c) Why does Zfone or SSH in the Baby Duck model not simply use a conventional authentication protocol like in SSL to authenticate the communication partners? What problem do they try to resolve?

Solution:

a)
Yes, since anyone can create a cryptographic identity (public key). It is not clear who (true identity) this entity is or what kind of access rights it has.

If it is sufficient to recognize a user again or to fight a man-in-the-middle attack between IDs, then a CA is not necessary.

b)
Since the knowledge of a key represents a way to identify entities and secure communication, one trusts that the other legitimate communication partner handles his keys in a responsible way.
If external entities like TTP or CA are used, one needs to trust their operation, that they send and forward the correct information. A CA or TTP is also used to connect key and identity for an entity. This can only happen if the entity can trust into the operation and accurateness of the CA or TTP. If this is not the case, one has to assume that maybe the key of the wrong person is sent or that confidentiality is not ensured.

c)
They assume that there is no global trustworthy TTP or CA, at least not one that they can use in all cases or for all application. Since security is almost impossible for the first contact without external help of CA or TTP, these protocols try to resolve the corresponding problems with other methods.

Task 5 Fighting Hotspots in Chord

From the lecture you should know two things. First, Chord proposes to replicate items to the k successors of the item ID for resilience. Second, this successor list as replica set cannot be hit by queries for the item. Thus, this does not help to fight hotspots (popular items are served by multiple nodes).

Modify Chord, so that all nodes in the replica set are utilized to answer queries for an item (Please note: This means that instead of using the successor list, do something different). Argue that your solution reaches this goal.

Solution:

Variant 1:

- Use root node (node holding the interval which fits to the item ID) and k predecessors for this instead
- When root node leaves, item has to be transferred to the successor of the last predecessor node.
- Regularily check if k predecessors hold the item and if you still are among the k closest predecessor
- Since all predecessors can be hit on the path to the root node, many queries will be answered by the predessesors. The root node will usually not be reached by queries since it can only be reached via its direct predecessor

Variant 2:

- Modify Chord so that k nodes share an ID and thus an interval with its items. All of these nodes are interconnected. In the routing table of other nodes, all of these nodes randomly appear, e.g. by always publishing all nodes in an interval in maintenance queries and the node selects one of them randomly for its finger (variant: bucket instead of one node per finger).
- Since the holders of the preceeding interval are always hit, for item balancing they should be updated more frequently (like in Chord Stabilization).
- Such a solution needs a maximum and minimum number of nodes that share an interval. Splitting and merging of intervals will become necessary.
- All nodes are hit by queries for their items if they are known by the predecessor nodes. For queries to other items, all nodes are hit if they are found in some routing tables.