# Home Assignement

Lorenzo Corneo (890225-1290)
Antonios Kouzoupis (890121-8837)
{corneo, antkou}@kth.se

May 29, 2015

## 1 System specification

Before showing the evaluation on the experimentations, it is necessary to explain briefly how we implemented the SWIM protocol and how we solved the problem of the NATs.

The aim of this report is not a full description of the system, but some knowledge is necessary to better understand the results we got in the evaluation phase.

### 1.1 SWIM

In this section it is explained what we did different than what was required from the specification of the assignment.

To begin with, it is required to piggyback information only during the ping but we strictly followed the SWIM paper and the propagation of such information happens also during the ping phase. In our opinion, this accelerate the convergence of the overlay as the changes are spread faster.

Then, when a pong is received, the peer selects up to `PIGGYBACK_SIZE` peers from the membership list, accordingly to its size, which have infection time smaller than the upper-bound. If the pong it is not received, the node sends an indirect-ping request to N random selected alive nodes.

Finally, as long as we believe that the random peer selection for the ping operation may not be optimal, we decided to implement a round-robin selection as suggested by the paper. A counter of pings is maintained and the selection happens between those peers with the lowest value. When a new peer joins the membership list, its counter is set to the minimum value (and not to zero) so that it will not be pinged many time sequently until it reaches the actual minimum.

### 1.2 SWIM through NATs

This section explains how we overcame the problem and how we propagate through the overlay changes in the parents list of the peers behind a NAT.

Peers behind NAT are not directly reachable so they must have relay peers which are open (they don't have NAT) and forward their traffic (in and out) through the network. The SWIM component propagates information also of the parents of peers behind NAT, so that other nodes know how to contact them. The NatComponent constantly sends heartbeats to the parents nodes in order to check whether they are still alive.

When the NatComponent discovers a parent is dead it gets new nodes from the CroupierComponent. The Croupier component returns a set of nodes that contains also dead nodes. For this reason, when the NatComponent receives this set, it queries the SwimComponent to check whether the nodes of the set are alive or dead. Up to an upper-bound of the alive nodes are set to be new parents for that node. Of course, the NatComponent can check itself whether the new candidate parents are alive, but this would add complexity to the component and, most of all, the SwimComponent already provides this service.

## 2 Evaluation

The assignment asks to retrieve the convergence time for different configuration, considering infection time, piggybacked information, number of peers and number of peers behind a NAT.

The SWIM protocol is highly configurable and, in particular, the most important parameters to configure are the infection time and the size of the piggybacked information. The authors of the paper suggest the infection time should be $\lambda log n$

This is not all, to guarantee the convergence in our experimentation we also varied the number of peers behind the NAT. For instance, if the number of those peers is too high the overlay will not converge.

### 2.1 Experimentation results

### 2.2 Consideration

We were not completely sure about the validity of those results because when we ran again the experimentations we got different results. Nonetheless, the theoretical studies done on SWIM made us able to know about the existence of an optimal configuration of the parameters.

After a deep analysis, we understood the gap between simulations on the same scenario was because we took timing in real-time. The Kompics framework utilises simulation time so that our results may be affected by the actual load of the CPU.

To solve this issue, we decided to implement a timer (with period 100ms) inside the AggregatorComponent which increments a variable accordingly to the time used by Kompics. Still, we don't obtain the real time but at least we get a consistent time.

## 3 Conclusion

The results we got are very meaningful because they made us understand that a underestimation of the parameters leads to the non convergence whilst an overestimation leads to a degradation of the convergence time.

Additional negative effects can be obtained, for example, when too many open nodes are killed at the same time so that it is impossible for the nodes behind a NAT to made them reachable by other peers.