**Self-Aggregation Techniques for Load Balancing in Distributed Systems**

1. **Introduction**

One of the today issues in software engineering is to find new effective ways to deal intelligently with the increasing complexity of distributed computing systems. In this context a crucial role is played by the balancing of the workload among all nodes in a system composed of interconnected heterogeneous nodes that enter and exit the system without following any rule.

**Assumptions:**
- Network of connected nodes.
- Nodes belong to a type (the network is not homogeneous).
- Nodes can only process and receive jobs of their own type.
- Jobs may arrive from other nodes or from the environment.

**Problem Description**
- Balance jobs of a given type among nodes of the same type.
- No centralization nor global network information is available.
- Nodes can appear and disappear unpredictably (churn).

**Nodes:**
- Nodes belong to a type (the network is not homogeneous);
- Heterogeneous nodes, where jobs cannot be processed indifferently by any of its neighbors (matchmaker);
- Nodes organize spontaneously into island of similar type at the same time.
- Generalize the algorithm to consider nodes that belong to more than a single type.

**Effective ways to deal intelligently with the increasing number of processed jobs with rewiring:**
- Load balancing iterations are inhibited by the fact that nodes are assumed to be homogeneous.

**Related Work on Self-Aggregation**
- Classical techniques include the work from Cybenko [4] and its extensions [2]. Another approach based on genetic algorithms [1]. Methods based on topology control in wireless multi-hop networks [3], nodes exploiting their local iterations. Examples: Nodes organize spontaneously into island of similar type at the same time. Generalize the algorithm to consider nodes that belong to more than a single type.

2. **Approach**

**Load balancing iterations are inhibited by the fact that nodes are assumed to be homogeneous.**

**Related Work on Load Balancing:**
- Classical techniques include the work from Cybenko [4] and its extensions [2]. Another approach based on genetic algorithms [1]. Methods based on topology control in wireless multi-hop networks [3], nodes exploiting their local iterations. Examples: Nodes organize spontaneously into island of similar type at the same time. Generalize the algorithm to consider nodes that belong to more than a single type.

3. **Existing Algorithms**

**Adaptive Self-Aggregation Algorithm:**
- A random node (initiator) asks one of its neighbors (matchmaker) for a node of the same type. If the node is found a new link is established between that node and the initiator, and a link is removed between that node and the matchmaker (see [5,7]).

**Dimension Exchange Algorithm:**
- In each iteration every couple of interconnected nodes will balance their job level (jobs are sent from the node that has more jobs to the node that has less jobs) [4].

**Simulation Environment:**
- Distributed Simulator written in Java.
- 20 Monte Carlo Simulations.
- Network of 100 nodes, average node degree of 4 neighbors.
- Scale-Free topology.
- Heterogeneity: 10% (10 types of nodes/jobs).
- Job distribution: static load of 400 jobs and continuous insertion of 400 jobs every 20s.
- Node processing time: a) 100% 5s, b) 70% 7s, c) 30% 3s (ideal throughput of 200 jobs/s).
- Node churn: every 10s 20% of the nodes disappear, and the same number appears.
- We evaluate the total number of completed jobs, the throughput, and the average number of messages exchanged by each node.

4. **Starting point: Self-Aggregation and Dimension Exchange Load Balancing**

**Self-Aggregation**

**Dimension Exchange Load Balancing**

5. **Combined Approach**

**1. We first apply one iteration of Load Balancing**

**2. We perform a partial clustering of the network with some further Load Balancing iterations**

**3. Finally we complete the clustering and we obtain the optimal function for Load Balancing**

6. **Simulation Results**

**Number of processed jobs without rewiring:**
- Load balancing iterations are inhibited by the fact that the jobs cannot traverse the nodes having different types.

**Number of processed jobs with rewiring:**
- We have a strong improvement with respect to the previous experiments.

**Number of processed jobs with rewiring and multiple bursts:**
- The curve grows as a straight line because of the continuous job bursts that are sent to the nodes.

**Throughput of processed jobs with rewiring and multiple bursts:**
- The throughput is close to the optimal value (the one obtained in homogeneous networks).

**Throughput of processed jobs in presence of a node churn (arrival/departure) of 10%:**
- The throughput is similar to the previous case, therefore it is resistant to this type of uncertainty.

7. **Conclusions**

**Generalize the algorithm to consider nodes that belong to more than a single type at the same time.**

**Study an efficient mechanism to dynamically adapt the frequency of load balancing and clustering iterations.**

**Consider the case in which the environment sends jobs of a given type to nodes of different types.**

8. **References**