Distributed Heartbeat Algorithm

Sandipan Dey,
CSEE Department,
University of Maryland, Baltimore County,
MD, USA,
sandip2@umbc.edu

January 22, 2010

1 Introduction

A heartbeat message is a periodical or regular message to a group of neighbors indicating that a process is still active and alive. A heartbeat algorithm works with heartbeat messages for each node, and is characterized by rhythmic expansion and contraction of information. Each node behaves like a beating heart (DCS Wiki).

The algorithm is a total algorithm and requires the presence of a globally synchronized clock. At each tick of the clock, each node sends the information it has to his neighbors and asks them for their information. In the first round, every node knows his neighbors only. In the second round, every node knows his neighbors and their neighbors. In each round the horizon is expanded by one hop. In the k^{th} round, every node knows about all k-reachable nodes from it, where k-reachability can be recursively defined as follows:

$$\Re_k\{n\} = \left\{ \begin{array}{ll} \{n\} & k = 0 \\ \Re_1\{\Re_{k-1}\{n\}\} & otherwise \end{array} \right\}$$

After D rounds, where D equals the diameter of the network, each node has informations about the whole system. The underlying assumption is that the network remains connected. The rest of this report describes the algorithm along with the experimental results and code fragment for the implementation of the algorithm.
For simulation purpose, random graph model $G(n, p)$ is used to generate the P2P network. Also, in the P2P network the basic assumption is that all peers work as servers as well as clients.

Algorithm 1 Distributed HeartBeat Algorithm

1: $\forall nodes$, declare local Variables (e.g., list of status for all nodes: each status being either of up, down or unknown).
2: initialize local variables.
3: **while** $(r < D)$ **do**
4: wait for central pulse generator or timer {synchronization}
5: send message to all neighbors {spread information (expansion)}
6: receive message from all neighbors {collect information (contraction)}
7: update local variables {add new information to existing ones}
8: $r \leftarrow r + 1$ {new round}
9: **end while**

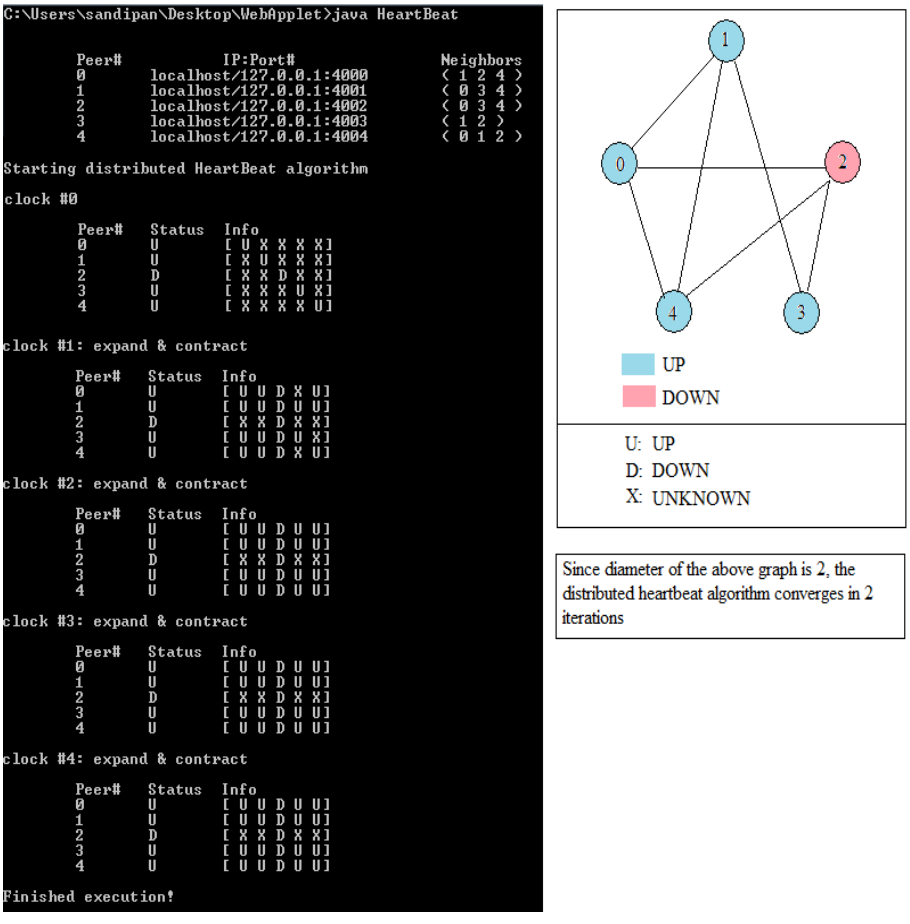

Figure 1: Status of the peers (nodes) for the Distributed HeartBeat Algorithm

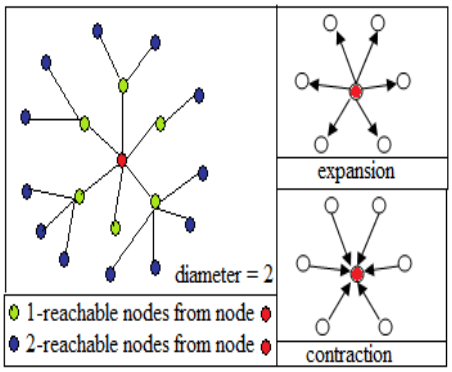

Figure 2: Expansion and Contraction Steps for the Distributed HeartBeat Algorithm

```java
public synchronized void doExpandContract(int clock) throws IOException
{
    for (int i = 0; i < adjPeers.size(); ++i)
    {
        Peer nb = (Peer)adjPeers.get(i);
        Socket socket =  new Socket(nb.ip, nb.port);
        out = new ObjectOutputStream(socket.getOutputStream());
        out.writeObject(new PeerMsg(PeerMsg.MsgType.INFORMATION, peersStatus, clock));
    }
}

public synchronized void updatePeersStatus(PeerStatus[] updatedStatus, int clock)
{
    for (int i = 0; i < peersStatus.length; ++i)
    {
        if ((peersStatus[i].status == PeerStatus.Status.UNKNOWN && updatedStatus[i].status != PeerStatus.Status.UNKNOWN)
            || updatedStatus[i].clock > peersStatus[i].clock)
        {
            peersStatus[i].status = updatedStatus[i].status;
            peersStatus[i].clock = clock;
        }
    }
}

public void runAlgo() throws IOException
{
    int n = peers.length;
    int clock = 0;
    for (int i = 0; i < n; ++i)
    {
        peers[i].startup(0);
    }
    peers[2].godown(0);
    System.out.println("clock #" + clock);
    printPeersStatus();

    for (clock = 1; clock <= n - 1; ++clock)
    {
        System.out.println("clock #" + clock + ": expand & contract");
        for (int i = 0; i < n; ++i)
        {
            peers[i].doExpandContract(clock);
        }
        printPeersStatus();
    }
}
```

```java
public class PeerStatus implements Serializable
{
    public enum Status
    {
        UNKNOWN, UP, DOWN;
    }

    Status status;
    int clock;
    PeerStatus(Status stat, int clk)
    {
        status = stat;
        clock = clk;
    }
}
```

Figure 3: Code fragment (Java) showing the expansion-contraction for the HeartBeat Algorithm