MST Construction in $O(\log \log n)$ Communication Rounds

Haneen Dabain

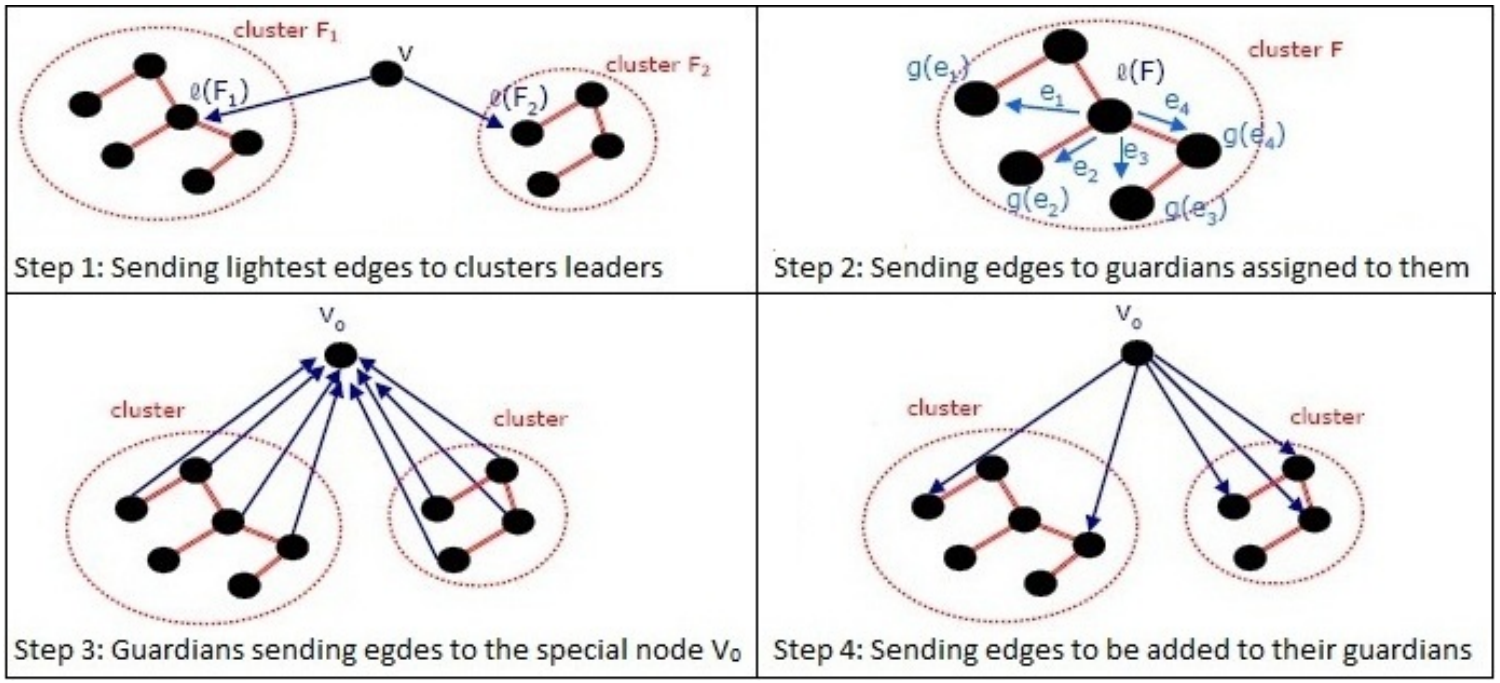
Department of Computer Science and Engineering
York University

March 10, 2010

Table of contents

Table of contents

The Algorithm



- V_0: A special node in the graph, e.g., the node with the smallest ID in the graph.
- $l(F)$: A leader of cluster F, e.g. the node with the smallest ID in the cluster.
- $g(e)$: A guardian node assigned to each minimum weight edge e.

Table of contents

Classes

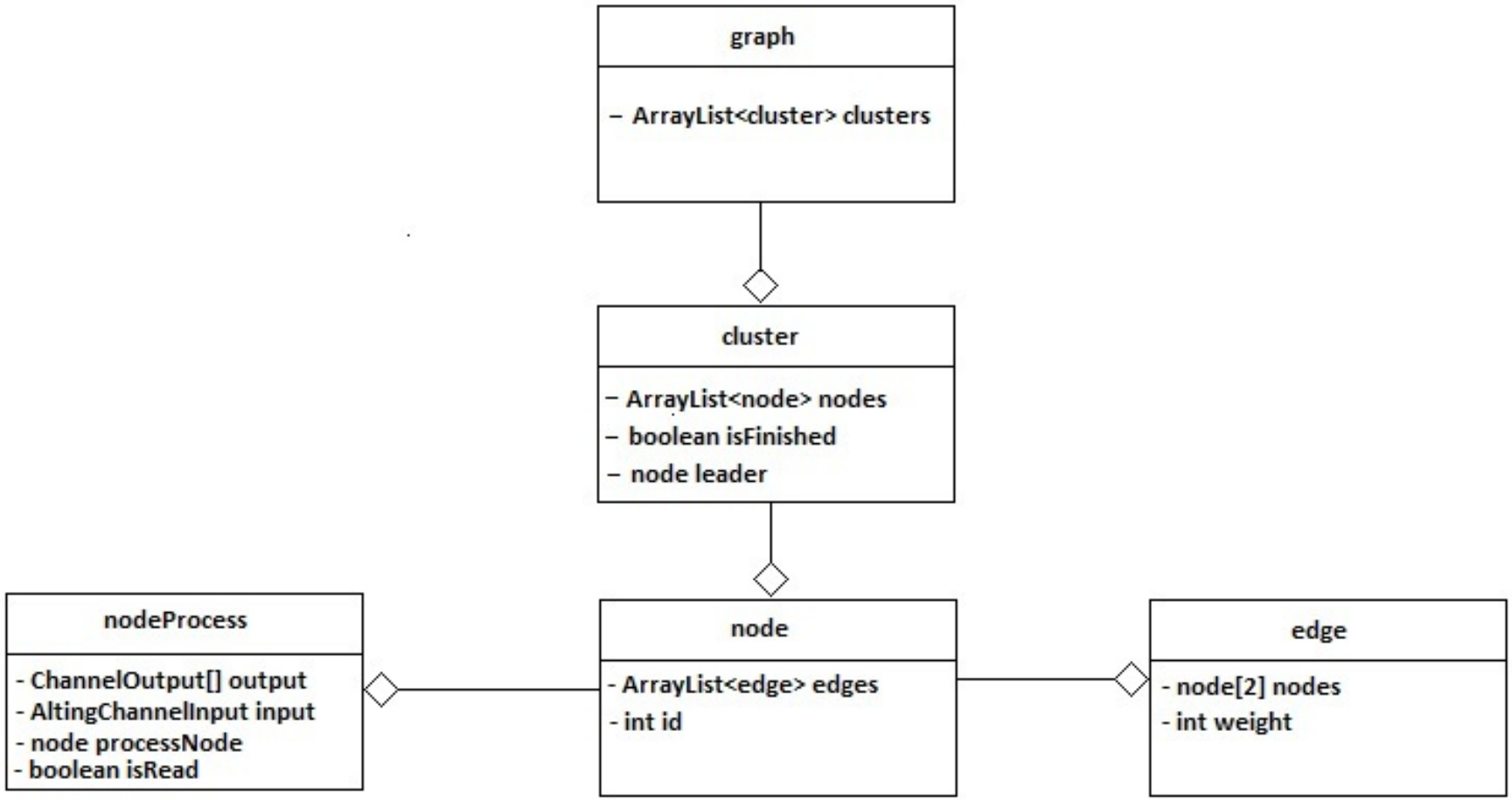
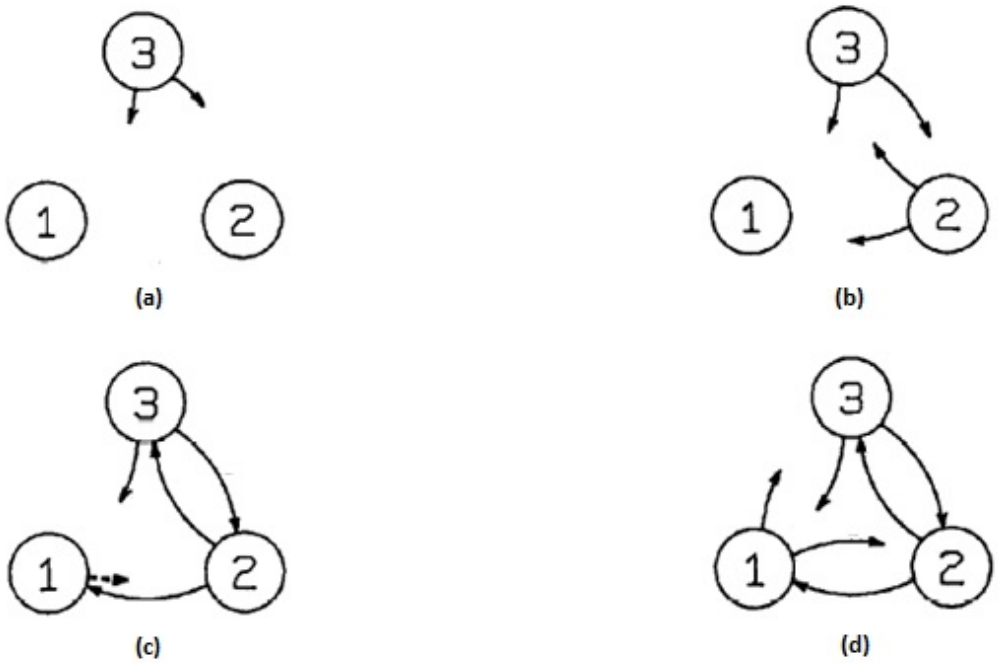

graph
– ArrayList<cluster> clusters
cluster
– ArrayList<node> nodes
– boolean isFinished
– node leader
nodeProcess
- ChannelOutput[] output
- AltingChannelInput input
- node processNode
- boolean isRead
node
- ArrayList<edge> edges
- int id
edge
- node[2] nodes
- int weight

Table of contents

The Model



- Nodes are considered as processes which are wired together using One2OneChannels
- Channels communication is synchronized.

Table of contents

Dead Lock

- Two processes executing a receive (read) command and waiting the other process to send (write)
- Both processes will be locked
- A single process can either send or receive but not both
- Each node has two processes running in parallel

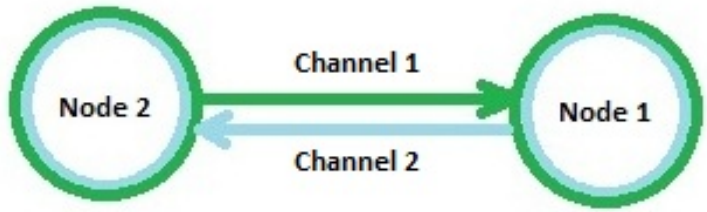


Table of contents

```
1   public class nodeProcess implements CSProcess{
2       private boolean isRead;
3       private ChannelOutput[] output;
4       private AltingChannelInput[] input;
5       private node processNode;
6       public nodeProcess(final ChannelOutput[] out) {
7           ....
8       }
9       public nodeProcess(final AltingChannelInput[] in) {
10          ....
11       }
12       public void run () {
13          if (isRead) {
14              ....
15          }
16          else{
17              ....
18          }
19   }
```

Table of contents

Communication Channels

final One2OneChannel[] inputOutputChannels = Channel.createOne2One(totalNumberOfChannel);

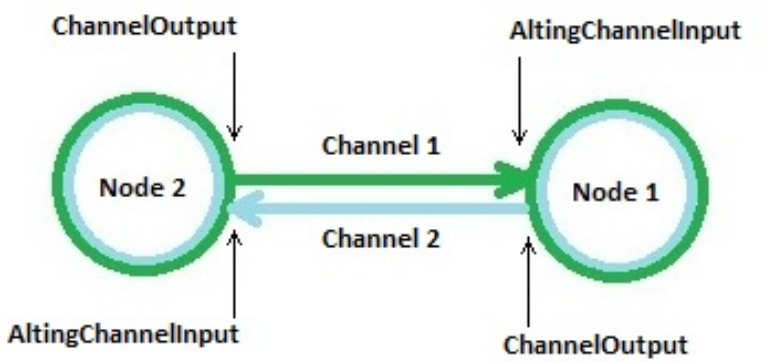
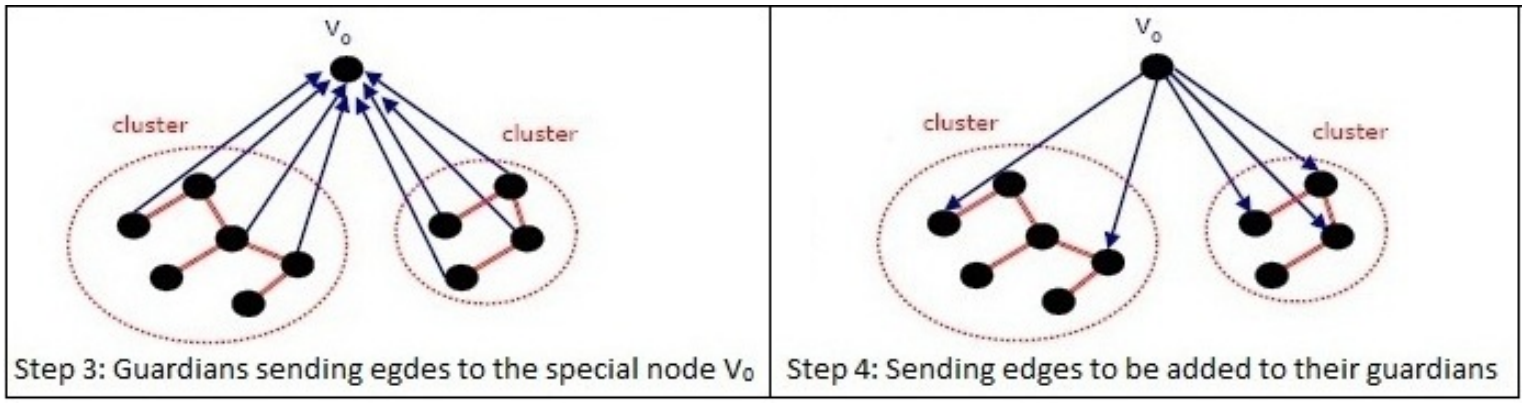


Table of contents

Parallelism




```
Parallel par = new Parallel;
CSProcess[] activeProcesses = new CSProcess[] {
        new Parallel(ActiveSendingProcs),
        new Parallel(ActiveReceivingProcs)}
par = new Parallel(activeProcesses);
par.run ();
par.releaseAllThreads();
```

Table of contents

Sending and Receiving

- Sending:
  output[index].write(message);
- Receiving:
  Alternative alt = new Alternative(input);
  int index;
  index= alt.fairSelect();
  input[index].read();

Table of contents

Our Implementation vs Sequential Implementation of an Algorithm by Gallagher, Humbles, and Spira

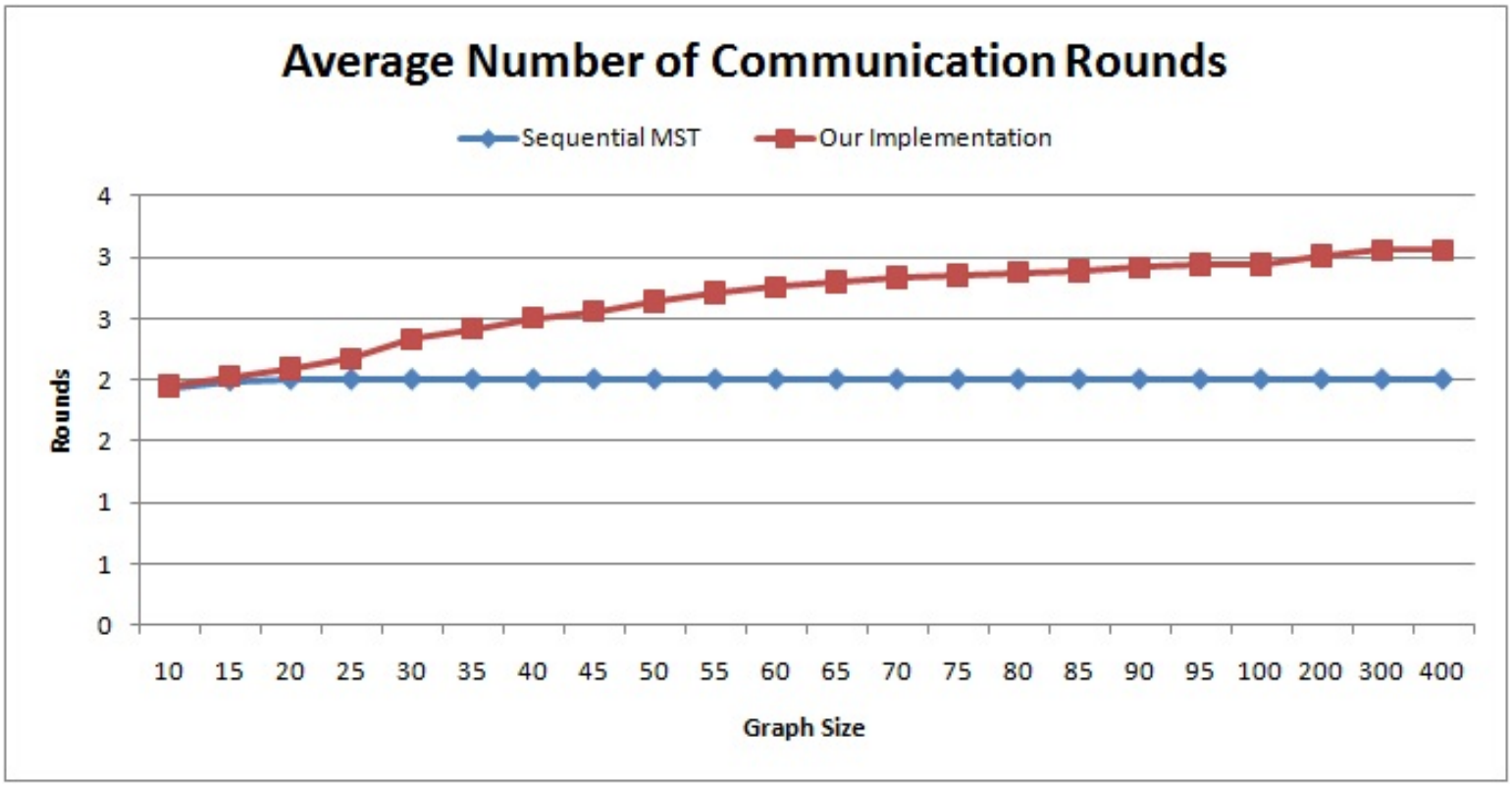
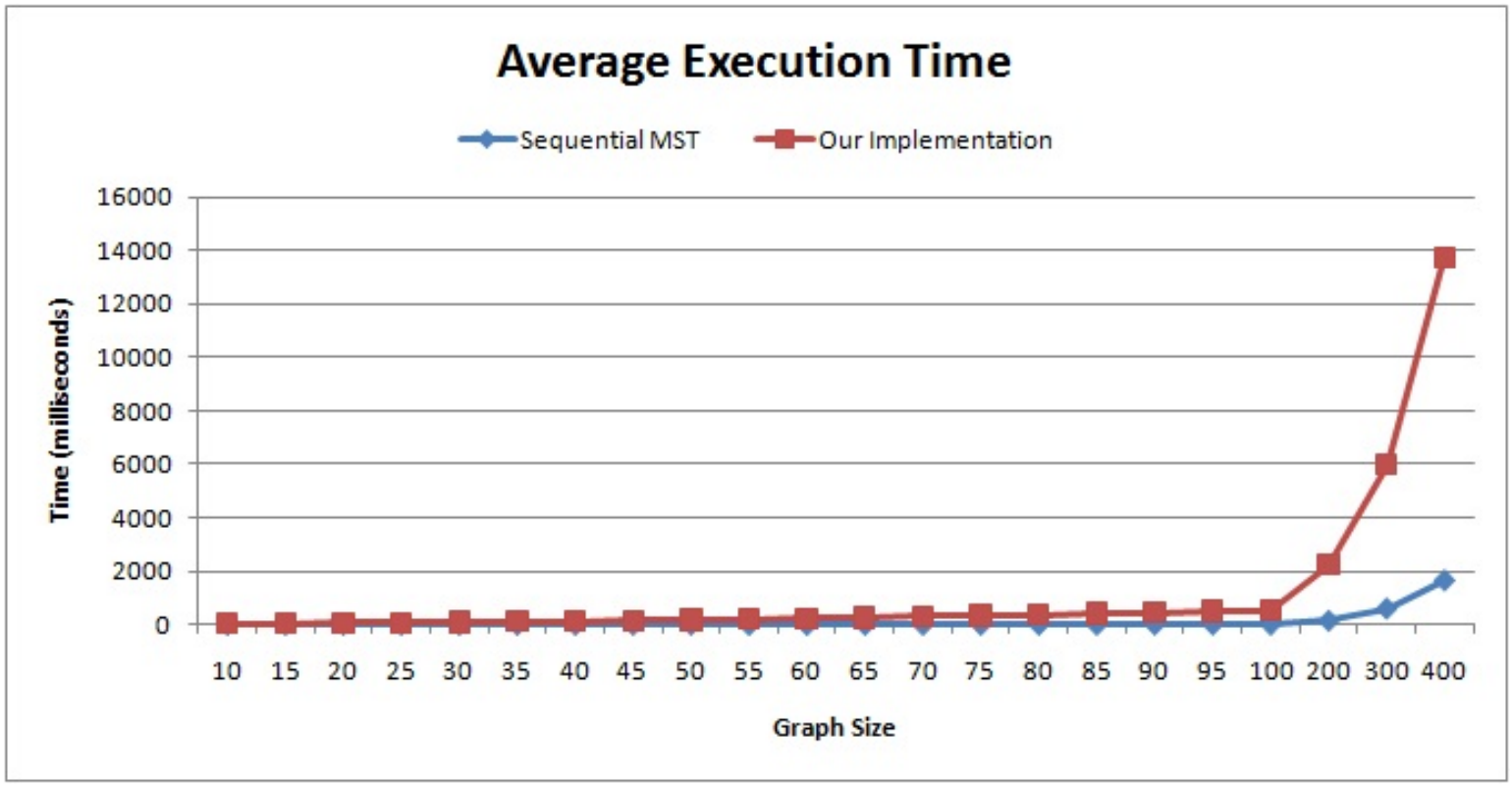


Table of contents

Our Implementation vs Sequential Implementation of an Algorithm by Gallagher, Humbles, and Spira



Possible Improvements

- Applying a simple synchronizer.
- Extension to handle messages of large size.
- Optimizing the implementation.