Performance of Modified OLSR Routing Protocols

Joel Frewin 21306458
Morgan Trench 21122107

The following is an analysis of the relative performance of a mobile ad-hoc network which implements modified versions of the OLSR protocol. in each experiment an altered version of the protocol is tested against the original in a standardised ns3 simulation environment. the simulation will see 50 nodes in a 1500x1500 area travel by the random waypoint mobility model at 1, 10 and 20 meters per second. this will be tested for 10, 20 and 30 nodes as sources of CBR traffic. in each simulation the output measured was the throughput of the entire network, specifically an average of the throughput to reduce noisy results and to better outline the trends. the following sections represent the four modifications made to the code.

Full 1-Hop MPR Sets
this protocol includes all 1-hop neighbours in the MPR set of each node. this was done by removing all of the conditional statements during the MPR set construction stage in the oils protocol. instead all neighbours are added. Lines 695 through to line 885 were commented out and replaced with:

```
for (NeighborSet::const_iterator neighbor = N.begin (); neighbor != N.end (); neighbor++)
  {
      mprSet.insert (neighbor->neighborMainAddr); // Add all of N
  }
```

Where N is the previously constructed set of all 1-Hop neighbors to a given node/interface. the relative results are shown in appendices 1.1,1.2, and 1.3 for node mobility speeds 1, 10 and 20 meters per second. this modification has resulted in slightly improved performance for all source node quantities and node speeds.

Halved MPR Sets
this protocol includes only half of the calculated members of a nodes MPR set by random choice. this was done by removing random MPR members from the completed set until it reaches the desired size. After line 885, just before what would have been the original mprSet is 'set', we reduce it by half randomly:

```
int desiredSize = mprSet.size()/2;
while (mprSet.size() != desiredSize)
  {
      MprSet::const_iterator iter = mprSet.begin();
      std::advance(iter, (rand() % mprSet.size()));
      mprSet.erase(iter);
  }
```

the relative results are shown in appendices 2.1,2.2, and 2.3 for node mobility speeds 1, 10 and 20 meters per second. this modification has resulted in reduced performance for all source node quantities and node speeds, with the performance gap closing as node speed increases.

Halved TC Messages
this protocol includes only half of a nodes MPR set in the topology control messages by random choice. this was done by removing random elements from the completed tables in the topology control messages until it reaches the desired size. This is done in much the same way
the previous part. After line 1216, we inserted the following:

```
int desiredSize = m_table.size()/2;
while (m_table.size() != desiredSize)
```

```
{
    std::map<Ipv4Address, RoutingTableEntry>::iterator iter = m_table.begin();
    std::advance(iter, (rand() % m_table.size()));
    m_table.erase(iter);
}
```

The relative results are shown in appendices 3.1,3.2, and 3.3 for node mobility speeds 1, 10 and 20 meters per second. this modification has resulted in significantly reduced performance for all source node quantities and node speeds.

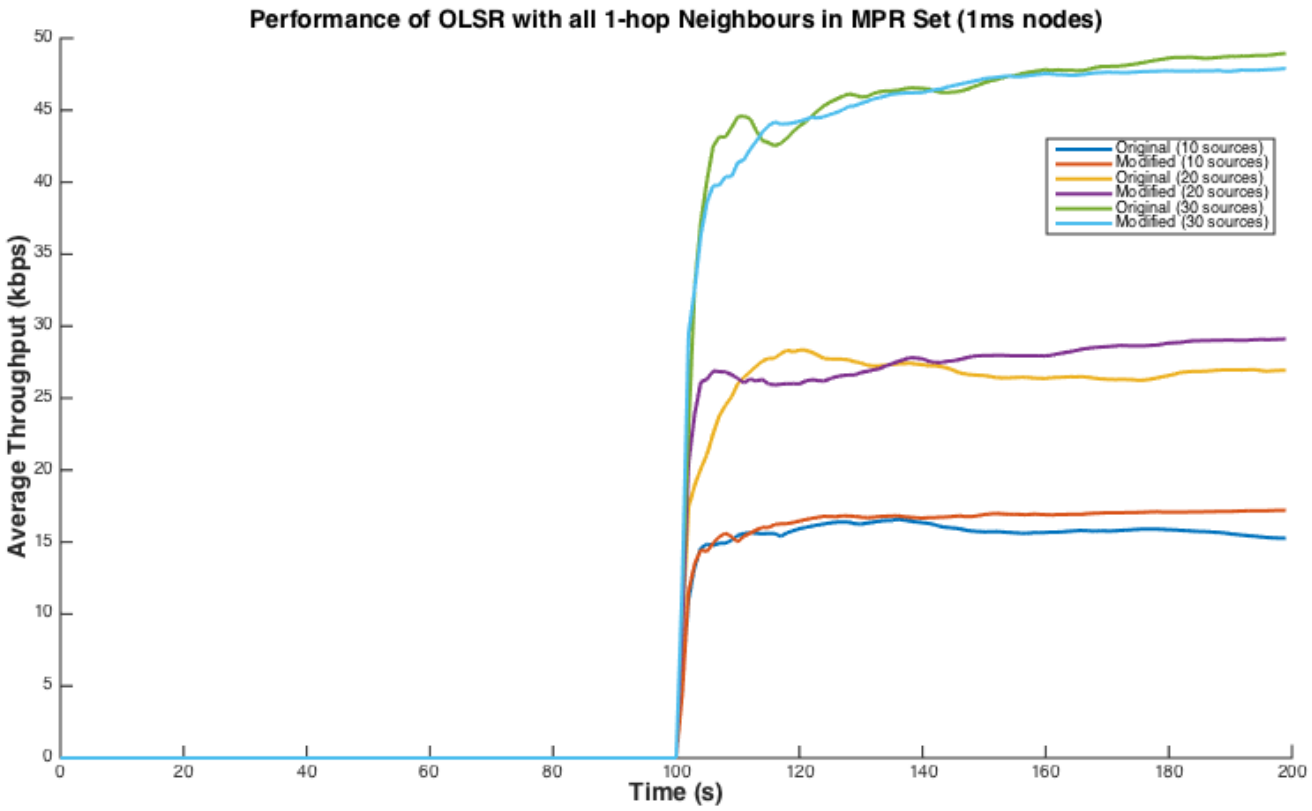
Non Bi-Directional Links
this protocol allows for the inclusion of nodes in the MPR set without bi-directional links. this was done by removing the checks that a tuple is symmetric in two places:

RoutingProtocol::MprComputation //line 609
RoutingProtocol::RoutingTableComputation //line 922

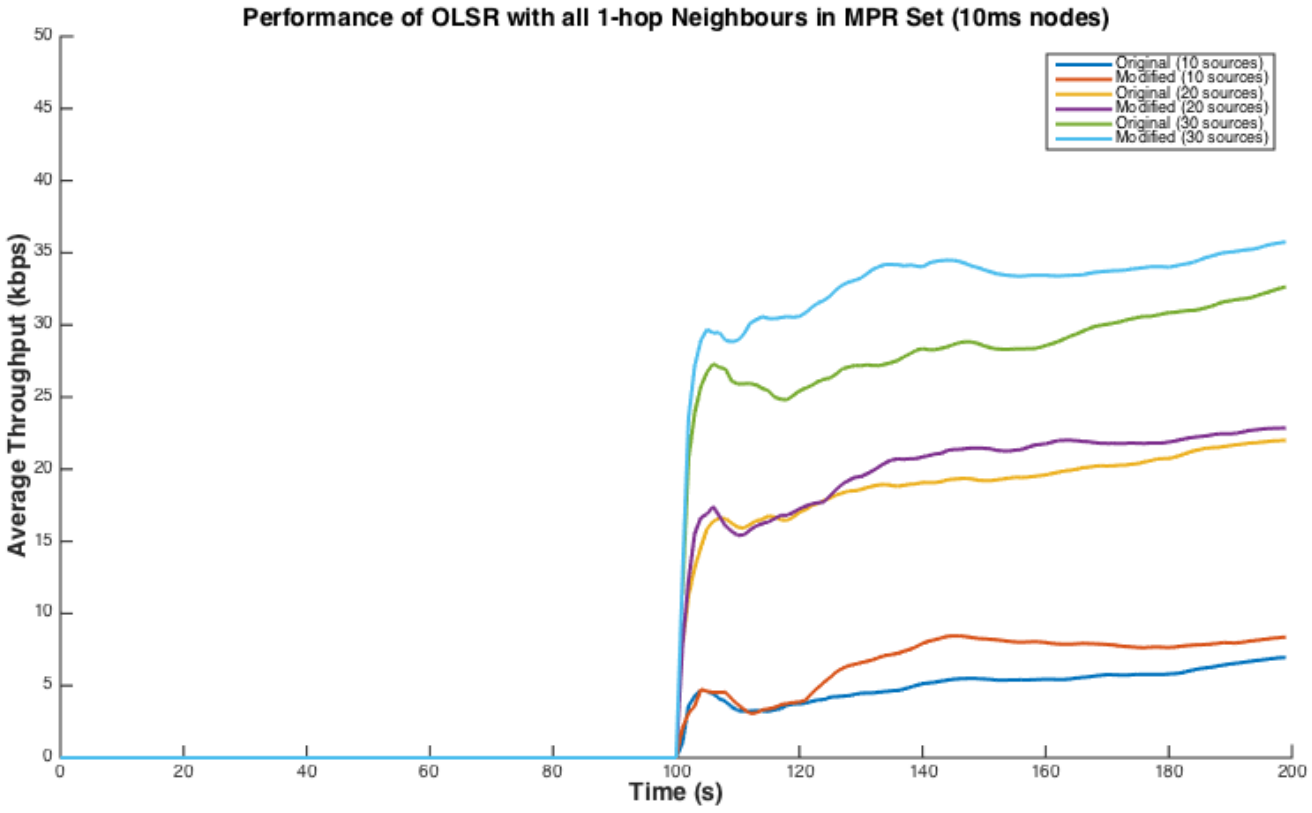
allowing modes with one way links to be added to MPR sets. The relative results are shown in appendices 4.1,4.2, and 4.3 for node mobility speeds 1, 10 and 20 meters per second. this modification has resulted in no significant change in performance for all source node quantities and node speeds.

Appendices

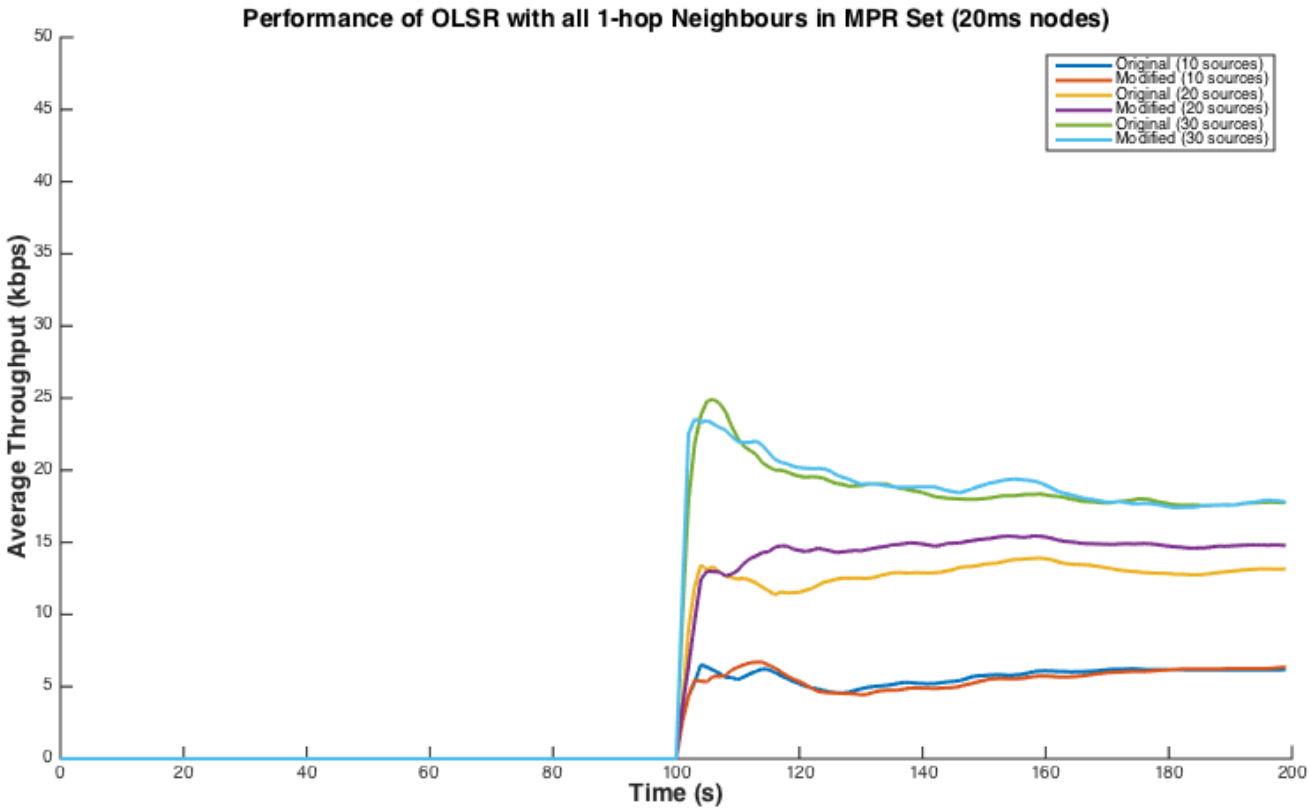
Appendix 1.1



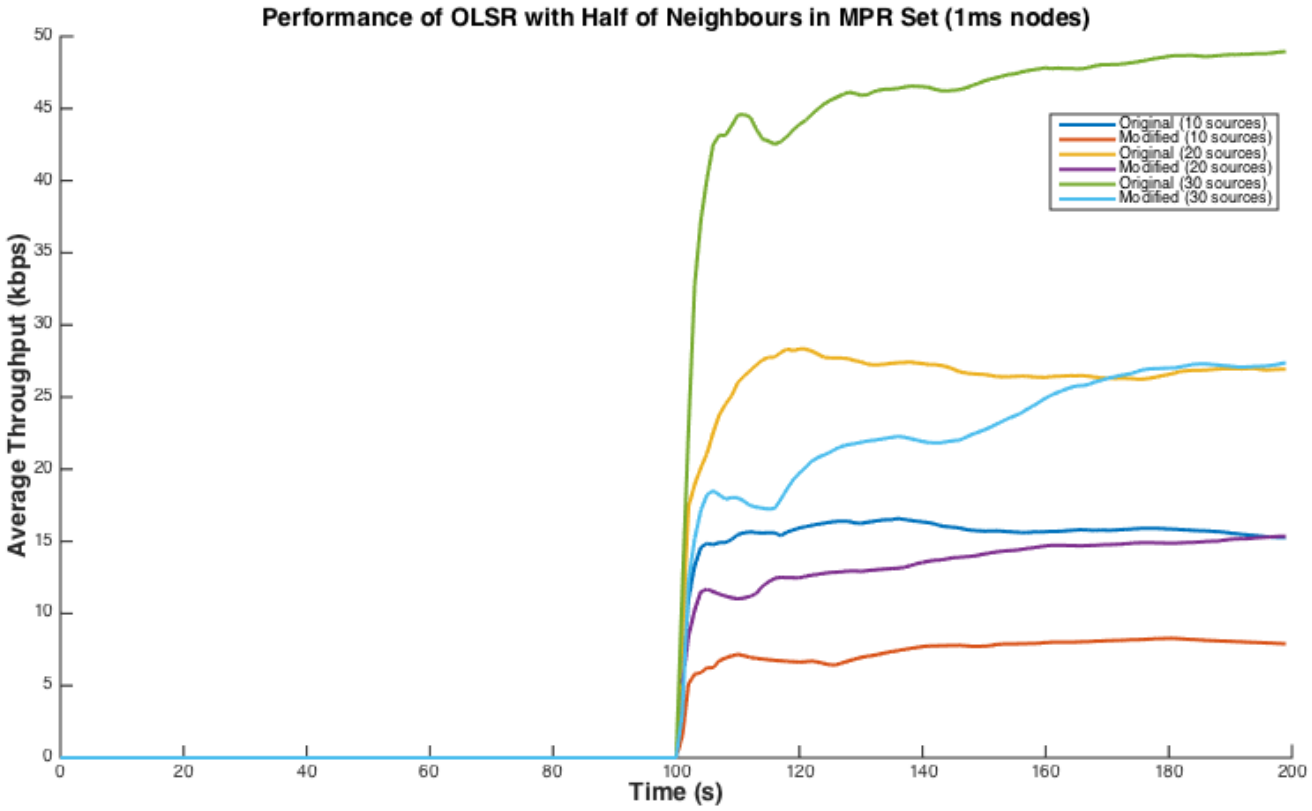
Appendix 1.2



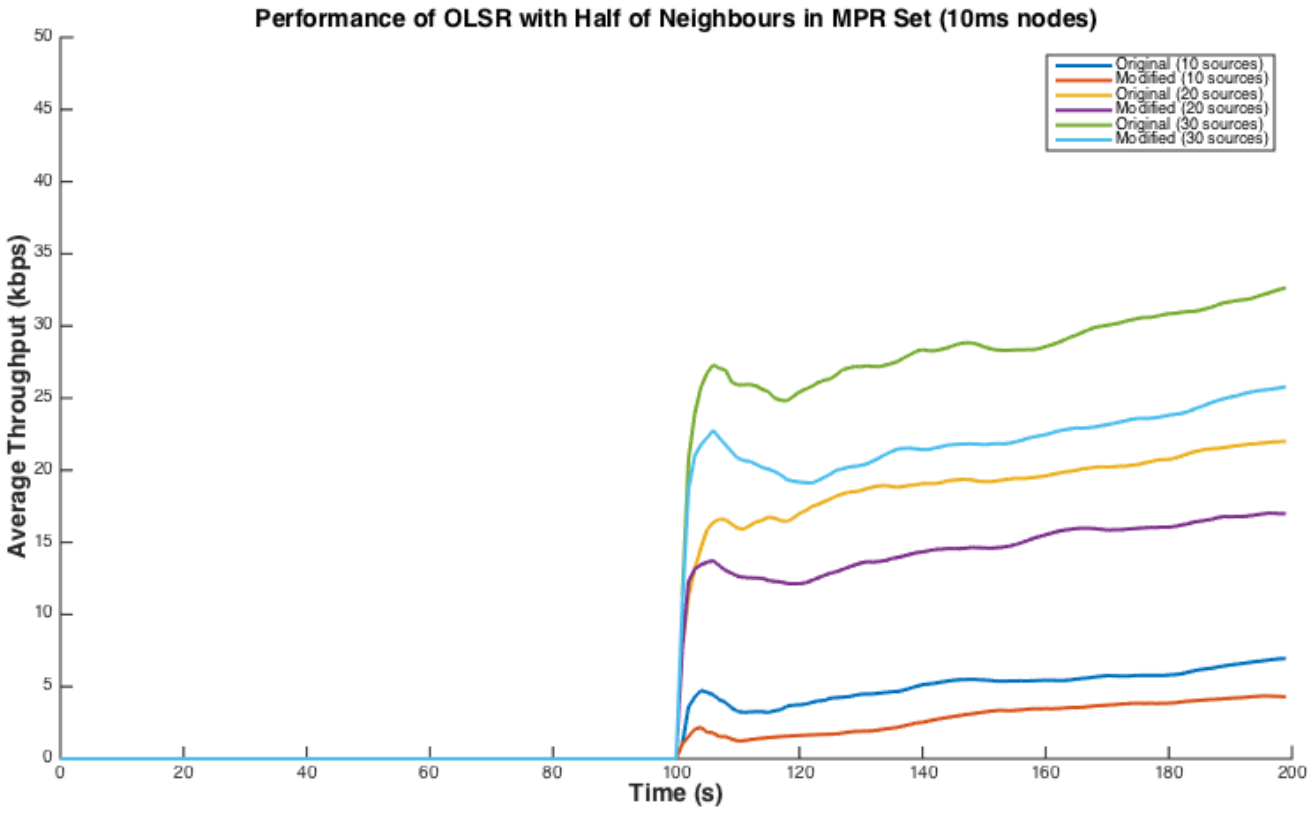
Appendix 1.3



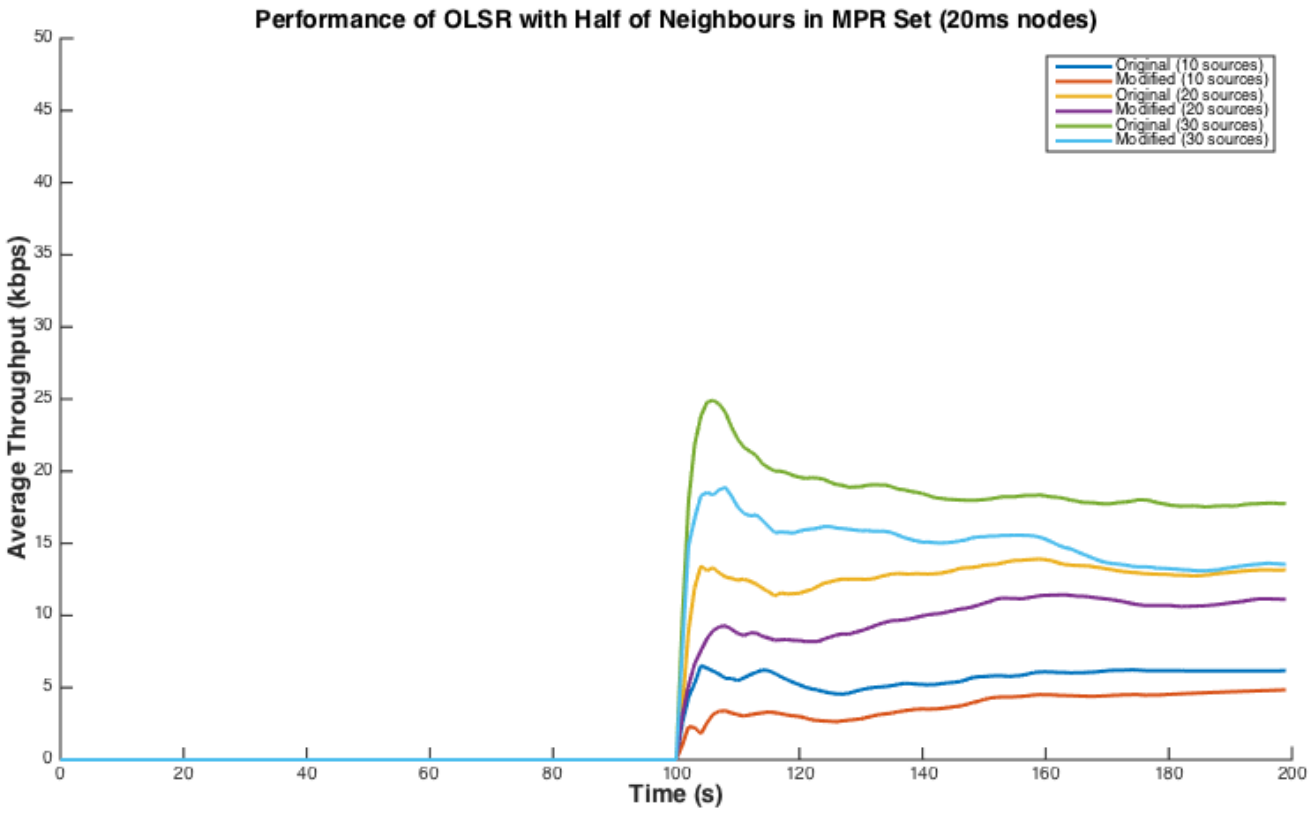
Appendix 2.1



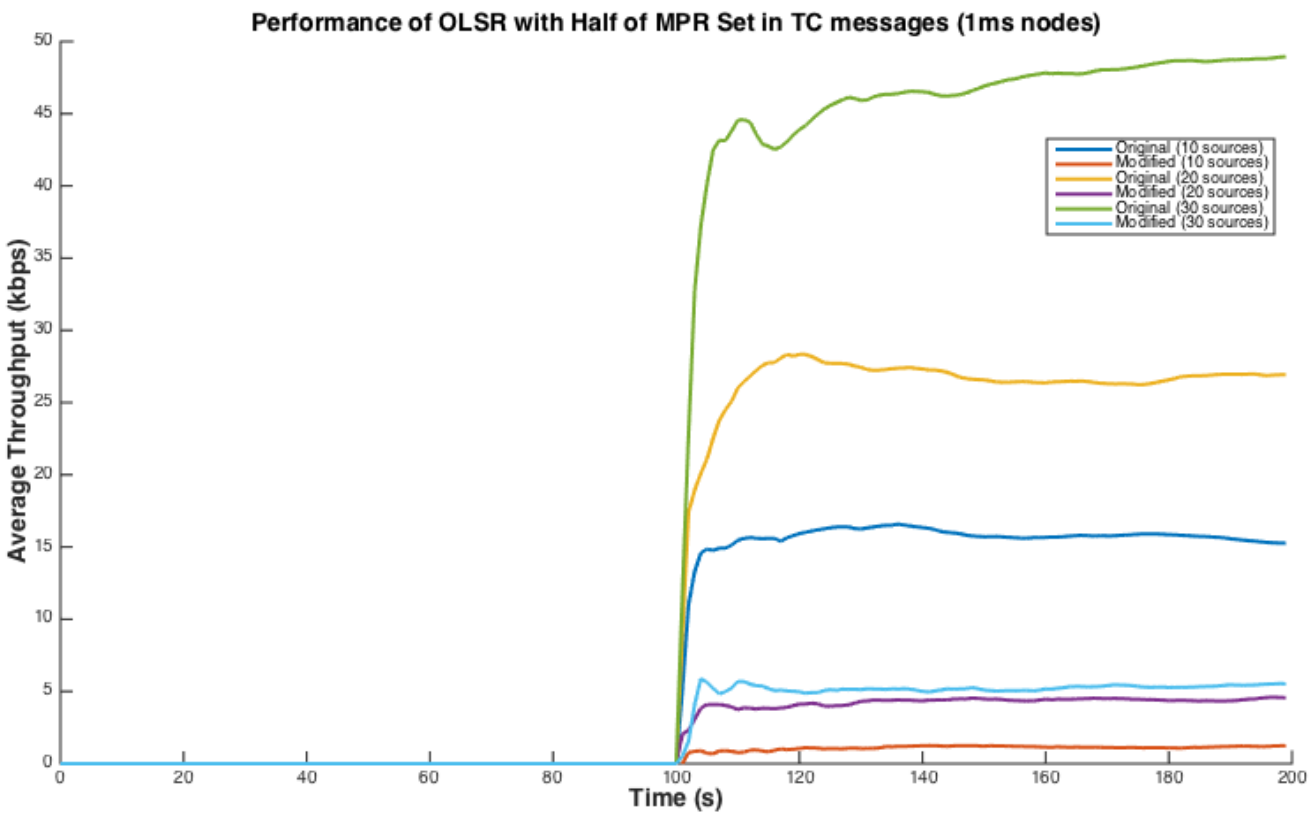
Appendix 2.2



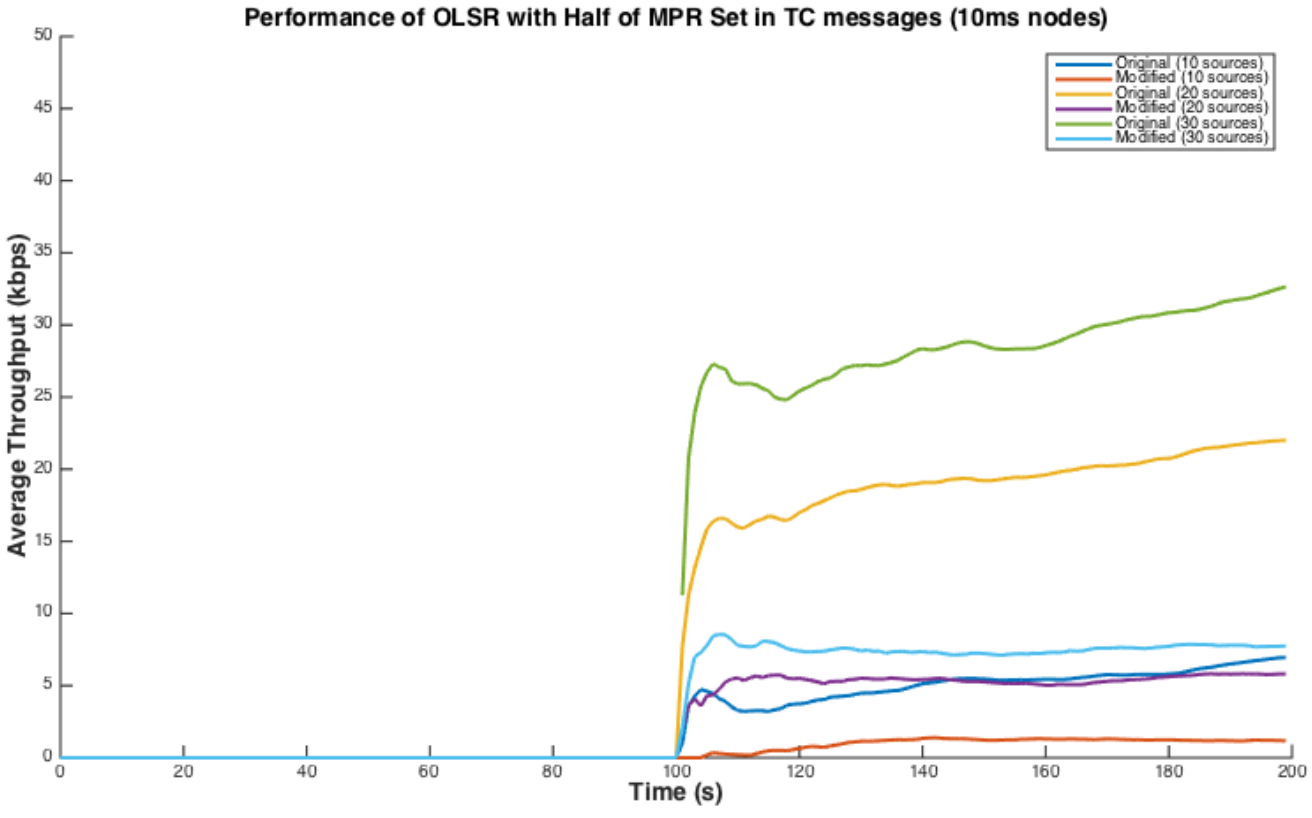
Appendix 2.3



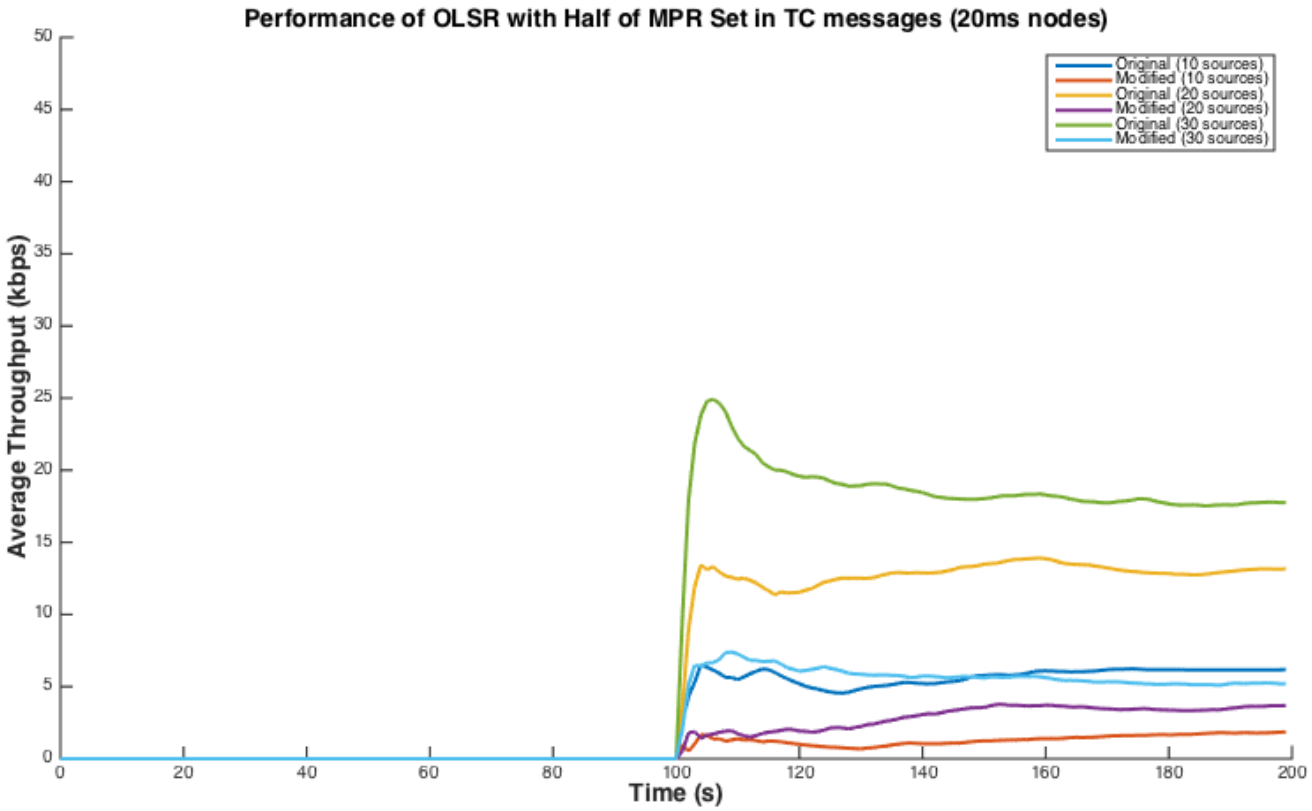
Appendix 3.1



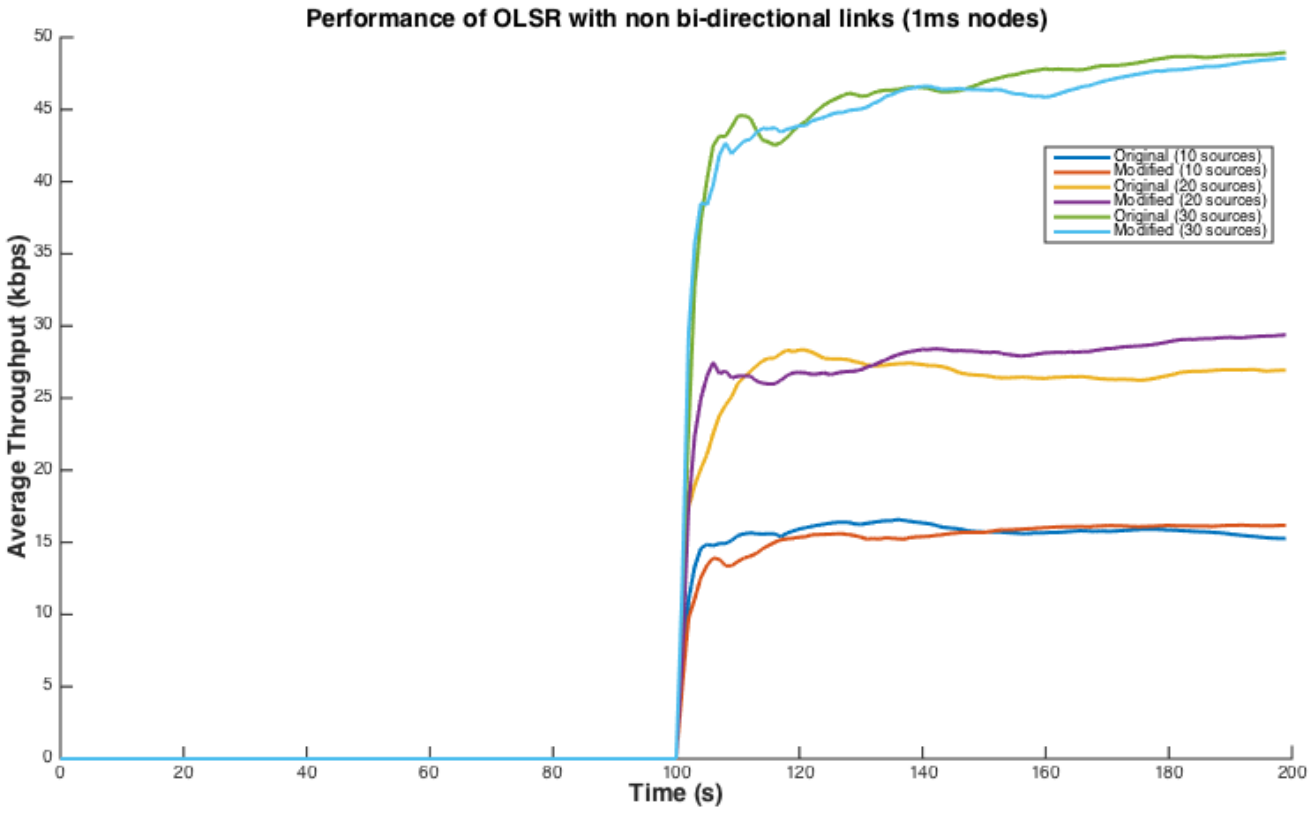
Appendix 3.2



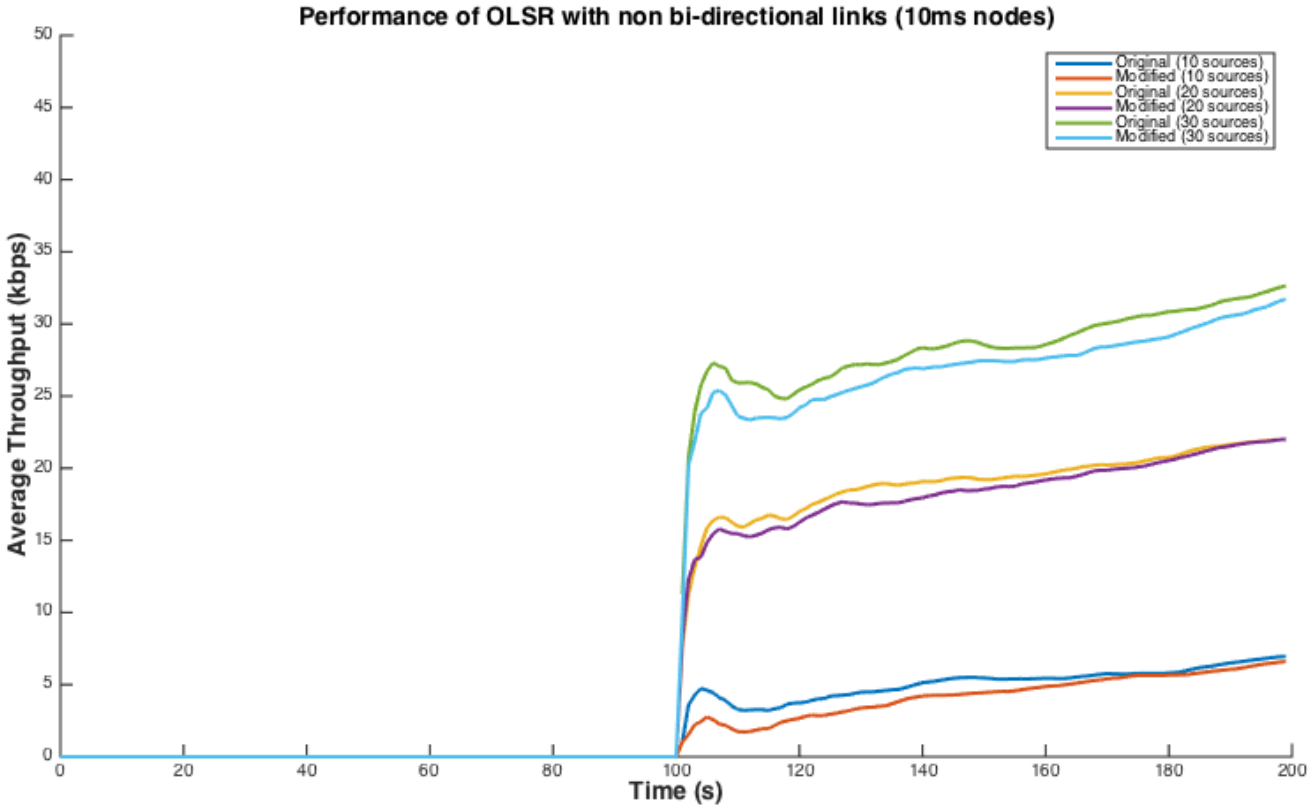
Appendix 3.3



Appendix 4.1



Appendix 4.2



Appendix 4.3