

In this program, two things have been done: (1) generate random topologies (2) performance test

(1) Generate random topologies

By using common used topology model of internet (Transit-Stub model) and Ad hoc (Locality model), we generate two kinds of network topology used for performance test.

Reference: E. Zegura, K. Calvert, and S. Bhattacharjee. "How to Model an Internetwork," In Proceedings of IEEE INFOCOM, April 1996.

(2) Performance test

In generated networks, selecting one node as source and some nodes as receivers in random (single source multicast network), we make performance test for network coding based multicast route algorithm (we put forward it ourselves, corresponding to high multicast rate and low multicast rate), shortest path distribution tree route algorithm (**DIJKSTRA** algorithm) and Maximum-rate distribution tree route algorithm (**PRIM** algorithm), respectively.

These performances are showed as follows.

- I) multicast capacity
- II) Load balance
- III) Resource consumption

Important functions Introduction

```
// the entrance of the whole program
void CMainFrame::OnGenerateTopology()

//used for generating Locality model topologies.
Graph * CGenerator::MakeGeoGraph(int edge_generate_method, long node_number, int scale,
                                double alpha, double beta, double gamma,
                                double min_capacity, double max_capacity,
                                double min_capacity2, double max_capacity2,
                                double radius1, double radius2)

// used for generating Transit-Stub model topologies.
Graph * CGenerator::MakeTsGraph()

//calculate the average degree of nodes
//return the maximum degree in the network
int CNetworkOperate::findDegreeDist(Graph *G, double * average_degree)

//get the maxflow between source_node and sink_node
double CNetworkOperate::MaxFlow(Vertex *node, int node_number,
                                int source_node, int sink_node, double * arcFlow)

//get multicast capacity of shortest path distribution tree route algorithm
//(method=DIJKSTRA) or Maximum-rate distribution tree route algorithm
//(method=PRIM)
double CNetworkOperate::GetMultiSessionCap(Graph * G, int method, int sourceNodeIndex,
int * receiveNodeIndex, int receiveNodeNumber)

//route by shortest path distribution tree route algorithm (method=DIJKSTRA) or
//Maximum-rate distribution tree route algorithm (method=PRIM)
BOOL CNetworkOperate::GetMultiSessionRoute(Graph * G, int method, double flow, int
sourceNodeIndex, int * receiveNodeIndex, int receiveNodeNumber)

//get the resources which have been consumed (by a multicast) in current graph G
double CNetworkOperate::GetUsedResource(Graph * G)

//route by network coding based multicast route algorithm
// when the multicast rate is high
double CNetworkOperate::HighRate_Coding_route(Graph * G, int sourceNode, int *
determineNode, int determineNum, double informationRate)

double CNetworkOperate::HighRate_Coding_adv_route(Graph * G, int sourceNode,
```

```
int * determineNode, int determineNum, double informationRate)
// when the multicast rate is low
double CNetworkOperate::LowRate_Coding_route(Graph * G, int sourceNode,
                                             int * determineNode, int determineNum,
                                             double informationRate, int pathNum)

double CNetworkOperate::LowRate_Coding_adv_route(Graph * G, int sourceNode,
int * determineNode, int determineNum, double informationRate, int pathNum)
```

P.S. Lacking of time, we have not translate Chinese comments of many functions into English comments. If you think the source codes is useful for you and can not understand some of them, then tell us and we will try our best to solve your problem.

Email: chikai@sohu.com kkchi@mail.xidian.edu.cn