Math 566 - Network Optimization

Final Project

Xiyu Xie

Fast Routing Algorithm

– Distributed Preflow Push for Maximal Flow

1. Motivation
2. Problem & Reduction
3. Distributed Requirement
4. Algorithms
5. Future Work

Motivation

Traditional routing algorithm is based on shortest path. A simple observation is, if there is additional independent path from source to destination, sending data using both path would be faster than using shortest path alone.

So shortest path routing is not the fastest routing algorithm. This project try to seek the mathematical model for fast routing, and implement it in the distributed fashion.

Problem

Given a connect network with link and node capacity limits, source propose to send certain amount of data destination as fast as possible.

That is, given amount of data, we want to send it in minimal time. This is equivalent to given a unit time, send as much data as possible. This is a maximal flow problem with transmission rate (bandwidth) as capacity.

Capacity limit on node could be transformed to capacity limit on edge by node splitting. So it could be reduce to a standard maximal flow problem with edge capacity only.

Distributed Requirement

Centralized algorithms we learned in class need to be split into the same fraction of code be implemented on each router.

1. No central controller, no priority assigned to node

That is, if a particular order of execution among node required, execution order is settle by negotiation among nodes (expect source).

2. Local information and communication

No node knows global information about the whole graph. It only know information about itself and could only communicate directly with its neighbors. It could get neighbors information through message passing, and store it locally.

3. Message delay and asynchronized algorithms

If assume once a node make a change, its neighbors know this change atomically, this is synchronized algorithms. In distributed system, synchronized is hard (deadlock) and expensive (a lot of communications).

Asychronized algorithms is more practical. That is acknowledge the existence of non-consistent data. This also keep node functional under suboptimal condition.

So the challenge in distributed algorithm design is reduce the need for sequential order and maintain final correctness in face of obsolete data.

Algorithms

Preflow push algorithms is less sequential than other path finding algorithms we learned in class, it may be a good candidate for routing.

Every node initiated with its own id, out and in neighbors and capacity of those links. And loaded the flowing code.

The source node started the program:

0. Preprocessing

Find number of node (n) on network : counting algorithms (not in reference)

Algorithm 1: Counting

```
source (s) send msg tag="counting" to its neighbors;
 if a node see it the first time,
        send to its own neighbors (except parent)
        wait for reply from neighbors
        reply sum(reply) +1 to parent
 else
        reply 0
```

* Inform destination (t) on the way

1. Initiate distance label:

Layered Depth First Search (Reversed from t)

(the one in reference not able to terminate)

Algorithm 2: Layered DFS

Destination iteratively increase search depth (T), until source is reached

msg: tag= “RDSF”, D= distance and T= depth (time to live field)

Node(i) recv msg

update distance;

decrement T by 1

if node is source reply n and finish

if T==0 reply its distance to parent

else

propagate msg to its neighbor (expect parent)

wait for reply from neighbors

reply its distance and whether finish to parent

Destination keep increasing T in each iteration until recv reply n

---

2. Preflow Push (See code for detail)

After the previous step, each node will have their neighbors information stored locally. Push and distance update (may not be valid) are all based on local information.

But when receiver get the push, it checks distance label and accept only from the one with higher distance and send accept message. Otherwise it reject and send updated distance label.

Sender update its local residual network and excess only after receiving accept reply.

Source and destination exchange their current flow information by some time interval. When the value match, it announce termination (and ask nodes to output flow, not needed in practice).

Future Work

1. Completely finish and polish code (Debug) (See code for detail)

   Initiation is hard coded. Termination is not implemented

2. Implement node splitting.

   Capacity on node may be more crucial in routing, and may be something meaningful out of this project. In that case, one code should take change of the pair of split nodes.

3. Think more about real world implement

   Read more about TCP conjunction control and multi-path routing. There may be some connections

4. General routing table

   Consider Pairwise case

   Compare overhead and memory requirement

Reference

General frame follows

"A Distributed Algorithm for the Maximum Flow Problem," Parallel and Distributed Computing, 2005. ISPDC 2005. The 4th International Symposium on , vol., no., pp.131,138, 4-6 July 2005 doi: 10.1109/ISPDC.2005.4

Minor change in Preflow push part.

Preprocessing and Initiate distance label are replaced by reading some online slide

ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-852j-distributed-algorithms-fall-2009/lecture-notes/