High-concurrency, High-throughput and Low-latency Computer Network Architecture Design

Haofei Yu 3180102099@zju.edu.cn
Yiheng Mao 3180103727@zju.edu.cn
Weiming Zhou * 3180105414@zju.edu.cn

Date: Version: 0.01

Last update: January 7, 2020

Abstract

This paper illustrates how to build the mathematical model of the architecture of a high-throughput and low-latency computer network. We abstract the problem of designing a computer network architecture into a graph model including topological information and edge weight information corresponding to the spatial information in real computer network architecture. By doing such transformation, we change the architecture design problem into a max number of shortest paths optimizing problem through adjusting the edge weights, which means we optimize the efficiency of the network by adjusting the location of the router. We think deeper into this model by 3 steps. First, we discuss the optimization problem of adjusting other routers in order to enhance the transmission efficiency between two fixed given vertices. Second, we adjust other routers to maximize the transmission efficiency between k source-target vertices pairs. Finally, we discuss how to get the best performance on the whole surface and how to change our adjusting policy when router distances are limited.

1 Introduction

1.1 Measurement of Computer Network

A Computer Architecture is a design in which all computers in a computer network are organized. A architecture defines how the computers should get connected to get the maximum advantages of a computer network such as better response time, security, scalability etc. The two most popular computer architectures are P2P (Peer to Peer) and Client-Server architecture. In reality. a well-designed computer network architecture must satisfy three features:

- **High-throughput**
  High-throughput usually means fast information transmission,which indicats that the ability of the network to handle huge information flow per time period is great. Network throughput is usually represented as an average and measured in bits per second (bps), or in some cases as data packets per second. Throughput is an important indicator of the performance and quality of a network connection. A high ratio of unsuccessful message delivery

*The lemmas and theorems are all done by ourselves without any reference because we do not find any graph problem is similar to ours, therefore we are willing to correct any mistake in our paper if anyone finds any mistake.

will ultimately lead to lower throughput and degraded performance.

The speed of uploading and downloading files is by default taken to mean the throughput of that particular internet connection.Therefore, throughput is a measurement for network connection speed.

- **High-concurrency**

  Both high-concurrency and high-throughput mean the ability to handle tremendous information flow at a short period, but the difference is that high-throughput needs the cooperations of several servers and high-concurrency needs the operation of a single server.

  High concurrency is one of the factors that must be considered in the architecture design of the Internet Distributed system, it usually means that the system can handle many requests simultaneously in parallel through the design.

- **Low-latency**

  Low latency describes a computer network that is optimized to process a very high volume of data messages with minimal delay (latency). These networks are designed to support operations that require near real-time access to rapidly changing data.

  While low latency and high bandwidth is the ideal to strive for, high latency has a deeper impact on load times than low bandwidth. At low latencies, data should transfer almost instantaneously and we shouldn't be able to notice a delay.

  We want to use math tools to help build a better network architecture.

1.2 Components of Computer Network

Evidently,if we want to go further into the network architecture, it is necessary to know the components of a computer network architecture.These components together makes it possible to transfer data from one device to another and makes smooth communication between two different devices.To be specific, a computer network includes Server, Client, NIC, Hub, Modem, Cable, Switch, Router.

Abstractly, a complete component in a computer network has the ability to receive information from particular components in the network and send information from particular components in the network at the same time, thus building connection with other components.

- **Client**

  Client plays a key role in sending the order and recieving the order to the server. It acts as the source and the target in a traditional computer network (Client-Server architecture).

- **Server**

  In Client-Server architecture, A server is a software or hardware device that accepts and responds to requests made over a network. Thus,together with client, server becomes the basic components of the computer network.

- **Router**

  General router can be considered to be the basis for internet connection. In general, a route refers to the path a data packet travels on a network. The route includes every device that handles the packet between its source to its destination, including routers, switches, and firewalls.

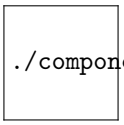

Figure 1: computer network components

1.3 Open Shortest Path First (OSPF)

Open Shortest Path First (OSPF) is a routing protocol for Internet Protocol (IP) networks. It uses a link state routing (LSR) algorithm and falls into the group of interior gateway protocols (IGPs), operating within a single autonomous system (AS).

In general, OSPF makes routers decide the flow of data packets based on **Dijkstra Algorithm** by updating and iterating a path list in the router.

Figure 2: The structure of OSPF

2 Model

2.1 Symbol Definition

Due to the computer network architecture, naturally, we choose the graph model with special limitation and special property as our model.The reality property and measurement have corresponding property and measurement in our built graph.

- **Graph** $< G(V, E) > or < K_n >$

  We use this graph symbol to represent the connectivity and unique topological structrue of the computer network architecture. Due to the connection is bidirectional, the graph model we use is definitely a undirected graph model.

  The node number is defined to be n, the edge number is defined to be m.To simplify our graph model, we only consider the complete graph model,indicating that we do an assumption that every component in a computer network is connected to any other components in the graph, so our low-latency problem can be simplified.

- **Vertex** $< V = \{v_1, v_2 ... v_n\} >$

  We use vertex on the graph to represent the components of the computer network. It can be client or server in traditional server-client architecture and can be any component in P2P architecture.

- **Edge** $< E = \{e_1, e_2 ... e_m\} >$

  We use edge to represent the connection between two vertices in the graph model and the corresponding meaning of the edge weight is time delay of the router in real computer network. As the edge weight represents the time delay, we regulate it to be positive.

- **Vertex Pair** $< (A, B) >$

  In the undirected graph $< G(V, E) >$,we can arbitaraily choose two vertices A and B(A is not the same as B), thus we can define a vertex pair for the preparation of the definition of Path.

- **Path** $< P_{vertex}(S,T) = \{S, V_1, ...V_k, T\} P_{edge} = \{e_0, e_1 ... e_m\} >$
  Given two vertices (S,T)as a vertex pair, we can define a path on the indirected graph from S to T, in which $V_0, V_1, ...V_k$ is the middle vertices in the path.
- **Length** $< L_{v_i,v_j} >$
  On the basis of the definition of path, we can define length, the length of a path is the sum of edges. We can define length from v_i to v_j by such a formula:

$$L_{v_i,v_j} = \sum_{e_k \in P_{edge}} e_k$$

- **Number of the shortest path** $< \psi(A,B) >$
  This is the core idea of the model. In one graph, we use the max number of shortest path between a vertex pair (A,B) to represent the number of shortest path. If there are paths between a pair vertex, the shortest path length is L_ψ, and ψ paths share the same path length L_ψ, then we can define the number of the shortest path to be M.
- **Number of the shortest path** $< \psi(A,B,n) >$
  This is the special form of the above symbol. It means the number of the shortest path between A, B in a complete graph K_n.
- **Sum of the shortest paths between any two vertices in a positive-edge-weight complete graph** K_n $< f(n) >$
  This $f(n)$ is what we want to prove in the P2P model.

$$f(n) = \sum_{i,j=0,1,2,...,n-1} \psi(v_i, v_j)$$

- **Sum of the shortest path number of n given vertices pairs which in the pair one is a fixed given vertex, the other is one of any other n vertices in** K_{n+1} $< g(n) >$
  This $f(n)$ is what we want to prove in the P2P model.

$$g(n) = \sum_{i=0,1,2...,n-1, v_i \neq t} \psi(v_i, t)$$

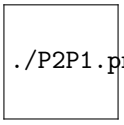

Figure 3: The Straightforward Understanding of $\psi(A,B,n)$, $f(n)$, $g(n)$

2.2 Model Target

The target of our model is to **reach the max number of the shortest paths by modifying the edge weights**.
The low-latency feature corresponds to the shortest paths in the complete graph K_n because the edge weight in our model represents the time delay while transmitting information, thus we guarantee the low-latency feature by sending the information through shortest path between two vertices.
The heigh-concurrency and high-throughput feature are guaranteed by maximizing the number of the shortest path. Both two features represents the capability of the network to deal with large amount of information, so we improve the network's ability to handle low-latency required requests and improve the robustness of the network when it comes to high-throughput low-latency requests. The procedure of sending requests from one client to another client is as follows:

- **Step1** One source client uses OSPF to get the shortest path to the target client in order to promise low-latency(low-latency guaranteed).
- **Step2** Our source client choose the policy that divides the data of the requests into several pieces and send them using the same length shortest path so that the requests will get to the target client at the same time and reduce the load of the whole computer network(high-concurrency guaranteed).
  In order to handle as much request as a client can, the client choose to divide the information into maximum pieces so that can deal with sequential information flow(high-throughput guaranteed)
- **Step3** Our target client received the partial requests from different routers and combine them by using special rules and information carried with the data.

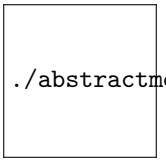

Figure 4: Abstract Model of Computer Network

To simplify the model, we assume that the only things that we can adjust is the time delay of the router, meaning adjusting the real location of routers.
Write the adjusting target in a mathematical form:

$$\begin{aligned} max \quad & \psi(A,B) \\ s.t. \quad & e_i > 0 \quad , \quad i = 1,2...m \end{aligned}$$

2.3 Three Different Models

- **Client-Server Architecture Model**
  Set one fixed vertex pair (S,T) , and we want to reach the max number of the shortest path by modifying edge weight.

$$\begin{aligned} max \quad & \psi(S,T) \\ s.t. \quad & e_i > 0 \quad , \quad i = 1,2...m \\ & S = T \\ & S,T \quad is \quad constant \end{aligned}$$

- **Peer-to-peer Network Model**
  Randomly choose one vertex pair (S_i,T_i) and (S_i,T_i) can both be chosen from V_1 to V_n , and we want to reach the max sum of number of the shortest path by modifying edge weight.

$$\begin{aligned} max \quad & \sum_{i=1}^{\frac{n(n-1)}{2}} \psi(S_i,T_i) \\ s.t. \quad & e_i > 0 \quad , \quad i = 1,2...m \\ & S_i,T_i \quad is \quad constant \quad i = 1,2...K \\ & S_i \neq T_i \end{aligned}$$

- **K-Client-Pair Model**

  Set K fixed vertex pair (S_i, T_i) and (S_i, T_i) are all different vertices,which is equivalent to choose $2K$ vertices and make pairs, and we want to reach the max sum of number of the shortest path by modifying edge weight.

$$max \quad \sum_{i=1}^{K} \psi(S_i, T_i)$$

$$s.t. \quad e_i > 0 \quad , \quad i = 1, 2...m$$

$$S_i, T_i \quad is \quad constant \quad i = 1, 2...K$$

$$S_i \neq T_i$$

$$K < n/2$$

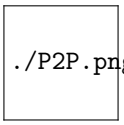

Figure 5: P2P model and Client-Server model

3 Model Proof

3.1 Client-Server Architecture Model

3.1.1 Lemmas and Deduction

Lemma 3.1. *Through all the shortest path from point S to T in a simple graph $G(V, E)$ with n vertices, there are no two paths go through two same points in opposite direction.*

Proof. If not, supposing there is a shortest path through i to j and another shortest path through j to i as Figure 1. The edge with arrow in the graph stand for how the paths go forward. In the path of blue color, marking the length of the two edges as K_1, K_2. Correspondingly, set the length of other two edges in the path of orange as L_1, L_2. Further more, K_3 and L_3 are both the sum of lengths from point j and i to the end point T.

Divide the situation into two parts. If $L_2 + L_3 > K_3$, then $L_1 + L_2 + L_3 > L_1 + K_3$, thus the latter is a shorter one from point j to point T so there is only one path from i to j, just as the lemma said.

On the contrary, if $L_2 + L_3 \leq K_3$, then $L_3 < K_3$ cause $L_2 > 0$ and $L_3 < K_2 + K_3$ naturally. So again there is $K_1 + K_2 + K_3 > K_1 + L_3$, the latter is a shorter way to the end. There is only one path from j to i, just as the lemma said.

In conclusion, if a shortest path contains i→j, there must be no other shortest path that contains j→i. □

Deduction From Lemma 1, after finding all the shortest paths from S to T, the simple graph $G(V, E)$ can be mapped to a directed graph $G' < V, E' >$, which contains all edges in shortest paths as E'. The edges can be marked with one-way arrow because there is only same-direction way between any two vertices among all the shortest paths.

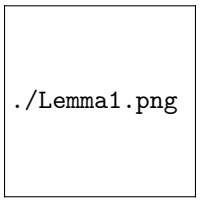

Figure 6: Portion of two hypothetical shortest path

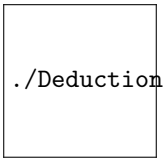

Figure 7: The mapped graph

Lemma 3.2. *(uniqueness) In the optimal solution (i.e. A set of assignment of edge weight that make the number of the shortest path in the graph be maximum), there must be a unique point that has the shortest distance to S among all the shortest path, and there is also one to T.*

Proof. Take T for example, and S is the same. Supposing there are several vertices $a_1, a_2, ..., a_k (k < n)$. From duduction, the corresponding directed graph $G' < V, E' >$ is used to explain. Let's take just the weight of edge from $a_i (i = 1, 2, ..., k)$ to T to be a one, and the weight of direct edges to $a_1, a_2, ..., a_k$ to be $L_1, L_2, ..., L_k$. (as in Figure 3) It is obvious that there is no edge between a_i and $a_j (i \neq j, i, j = 1, 2, ..., k)$ because $a_i \rightarrow a_j \rightarrow T$ is clearly not in a shortest path.$(1 + L_{ij} > 1)$

However, a new set of assignment can be given to generate more shortest paths. As is shown in Figure 4, the length between a_i and T is reset to be $[1 - (i - 1)\epsilon]$; $L_1, L_2, ..., L_k$ are reset as $L_1, L_2 + \epsilon, ..., L_k + (k - 1)\epsilon$. Adding $(k - 1) \times k/2$ edges between $a_i, a_j (i \neq j, i, j = 1, 2, ..., k)$, and setting the length of edge $\{a_i, a_j\}$ as $|i - j| \times \epsilon$. Therefore, the original shortest paths are still the shortest, and in addition there are much more than before. For example, $L_1 + \epsilon + (1 - \epsilon) = L_1 + 1$.

Thus, a contradiction arises as the optimal hypothesis is no longer right. □

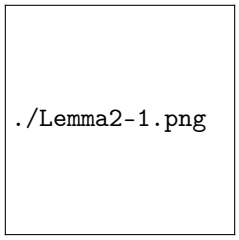

Figure 8: The hypothetical situation

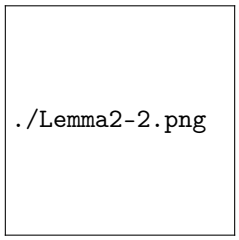

Figure 9: A new set of assignment

3.1.2 Special Cases

n=2 There is only one edge so the maximum number of shortest path is 1, too.

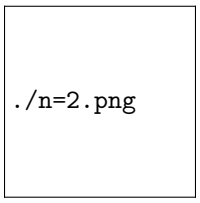

Figure 10: Case n=2

n=3 Set $L_{13} = 2, L_{12} = 1, L_{23} = 1$, and then the maximum number of shortest paths is 2.

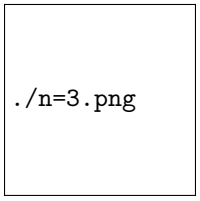

Figure 11: Case n=3

3.1.3 Conclusion of Client-Server Architecture Model

Theorem 3.3.

$$\psi(S,T,n) = 2^n, \quad S,T \quad is \quad fixed \quad points$$

Proof. From the lemma above, there exist a unique point t' such that t' is the nearest point from t . Now we divide all the shortest paths from s to t into two types. The shortest path in type one contains t', and will not go to other point rather than t as t' is the unique nearest point , which means these shortest paths(tt' removed) are also the shortest paths to t'. From the definition of $f(n)$ we can obtain the number of type one shortest paths will not be greater than $f(n-1)$. From definition, the type two shortest path won't pass point t' , so the number of type two shortest paths will not be greater than $f(n-1)$. So the total number of shortest paths is no greater than $2f(n-1)$. So we get:

$$f(n) \leq 2f(n-1)$$

As $f(0) = 1, f(1) = 2$, we can obtain $f(n) \leq 2^n$. Now we can construct the graph which has 2^n shortest paths from s to t. Label s as 0,t as $n + 1$, other from 1 to n. The distance of (u, v) is $|u - v|$, where u, b are labels. It is easy to see the number of shortest paths from s to t is 2^n. □

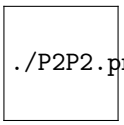

Figure 12: The Straightforward Understanding of the Proof

3.2 Peer-to-peer Network Model

3.2.1 Analysis

Every dot pair (u, v) in a complete graph G_1 with positive edge weight can be mapped to a directed graph G_2 with $\frac{n^2-n}{2}$ dots:

If a dot pair (u_1, v_1) is contained in one of another dot pair (u_2, v_2)'s shortest paths, then there is a directed edge from point A to B in G_2, where A represents (u_2, v_2), and B represents (u_1, v_1).

It is obvious that there is no cycle in G_2 cause if there is $A \rightarrow B$ in G_2, there must be $L(u_2, v_2) < L(u_1, v_1)$ in G_1.Thus, G_2 is a DAG.

3.2.2 Lemma

Lemma 3.4. *In all the shortset paths in G_1, there must be at least one point to be none of each path's intermediate points.*

Proof. As what has been put forward, G_2 is a DAG, then there must be a point P with indegree of 0. Suppoing it is corresponding to a dot pair (s, t) in G_1, then at most one of s and t would be a intermediate point in any shortest paths. Otherwise, if s is in any of $L(p, q)$, t is in any of $L(u, v)$, then $p \rightarrow ... \rightarrow s \rightarrow ... \rightarrow q \rightarrow ... \rightarrow u \rightarrow ... \rightarrow t \rightarrow ... \rightarrow v$ is a shortset path. Naturally, Q(represents (p, v)) points to P in G_2, which is a contradiction. □

Lemma 3.5. $f(n) \leq f(n-1) + g(n-1)$.

Proof. From Lemma 1, there must be one point u in G_1, which would not be any shortest paths' intermediate points. So the number of sum of shortest paths in $f(n)$ can be diveded into TWO parts:

1. $g(n-1) = \sum_{v \in G_1, v \neq u} (Number of L(v, u))$.
2. $f(n-1) = \sum_{v,m \in G_1, v \neq m, v, m \neq u} (Number of L(v, m))$.

Therefore, $f(n) \leq f(n-1) + g(n-1)$.

□

Lemma 3.6. $g(n) \leq g(n-1) + h(n-1)$

Proof. Thinking about sum of number of shortest paths from n points to point t. Then as deducted in PROBLEM 1, just one direction can be determined for every edge in each shortest path, also there is no cycle. So the new directed graph is a DAG, too. A point k with indegree of 0 exists, which means $g(n)$ can be diveded into TWO parts:

1. $h(n-1) = Number of L(k,t)$.

2. $g(n-1) = \sum_{v \in G_1, v \neq k, t}(Number of L(v,t))$.

Therefore, $g(n) \leq g(n-1) + h(n-1)$. □

3.2.3 Conclusion of Peer-to-peer Model

Theorem 3.7.

$$f(n) = 2^n - n - 1$$

Proof. Due to the known truth that $f(1) = 0, g(1) = 1$ and above lemma, we can make an induction that $f(n) \leq 2^n - n - 1$. Moreover, we can construct a complete graph K_n(in that graph , the label of s is 0 and the label of t is (n-1),other points we use the label from 1 to n-2).The edge weight of the edge between u and v is $L_{u,v}$,then the sum of the shortest paths of any two vertices is $\sum_{i=0}^{n-2}(n-1-i)2^i = 2^n - n - 1$, therefore $f(n) = 2^n - n - 1$ is proved. □

4 Conclusion

The whole paper discusses two basic models which fit the P2P Network Model and the Client-Server Network Model. Further work can be done to get better models to fit the ideal architecture of the computer network.