International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 04 Issue: 11 | Nov -2017 www.irjet.net p-ISSN: 2395-0072

Understanding network routing problem and study of routing algorithms and heuristics through implementation

Saumya Shandilya

Saumya Shandilya, Computer Science Department, Symbiosis Institute of Technology, Pune.

Abstract - *In this project, we intend to identify, understand and compare various routing algorithms used in real world networks.*

The various objectives of this research are :

1. *Define and understand the concepts of routing.*
2. *Determine if a Greedy or Dynamic Programming strategy algorithm is more efficient for routing, in general. Identify which strategy is used more in real world networks.*
3. *Identify the common routing algorithms used in networks. Identify which algorithms are used in which scenarios.*
4. *Identify the performance metrics for gauging algorithms.*
5. *Compare existing routing algorithms in various scenarios (on the simulation software). Also note specific phenomena or anomalies during simulation.*
6. *Think of modifications (if any) in existing routing algorithms, or devise a new routing algorithm.*
7.

Key Words: Routing, Throughput, Latency, Greedy Strategy, Dynamic Programming

1. INTRODUCTION

The transport layer provides communication service between two processes running on two different hosts. In order to provide this service, the transport layer relies on the services of the network layer, which provides a communication service between hosts. In particular, the network-layer moves transport-layer segments from one host to another. At the sending host, the transport layer segment is passed to the network layer. In order to this, the network layer requires the coordination of each and every host and router in the network. In simple terms, if we have to define Routing in a lay man's language we can simply say that Routing is the manner/order in which we decide the path a segment shall follow from the sending host to the receiving one. This path includes a connection of links and routers. In technical terms though routing is a complex yet challenging concept.

Technically, Routing broadly consists of the following 3 functions:

1. Path Determination: This function determines the path/route the packets will follow from the sender to receiver. It involves various routing algorithms which are discussed further.
2. Switching: When a packet arrives at a router it needs to be further dispatched to other routers i.e. it is further switched to other routers.
3. Call Setup: Just like a TCP carries out 3 -way handshake similarly some network layer architectures (e.g., ATM) requires that the routers along the chosen path from source to destination handshake with each other in order to setup state before data actually begins to flow. In the network layer, this process is referred to as call setup.

The main goals of routing are:

Correctness: The routing should be done properly and correctly so that the packets may reach their proper destination.

Simplicity: The routing should be done in a simple manner so that the overhead is as low as possible. With increasing complexity of the routing algorithms the overhead also increases.

Robustness: Once a major network becomes operative, it may be expected to run continuously for years without any failures. The algorithms designed for routing should be robust enough to handle hardware and software failures and should be able to cope with changes in the topology and traffic without requiring all jobs in all hosts to be aborted and the network rebooted every time some router goes down.

Stability: The routing algorithms should be stable under all possible conditions.

Fairness: Every node connected to the network should get a fair chance of transmitting their packets. This is generally done on a first come first serve basis.

Optimality: The routing algorithms should be optimal in terms of throughput and minimizing mean packet delays.

Here there is a trade-off and one has to choose depending on his suitability.

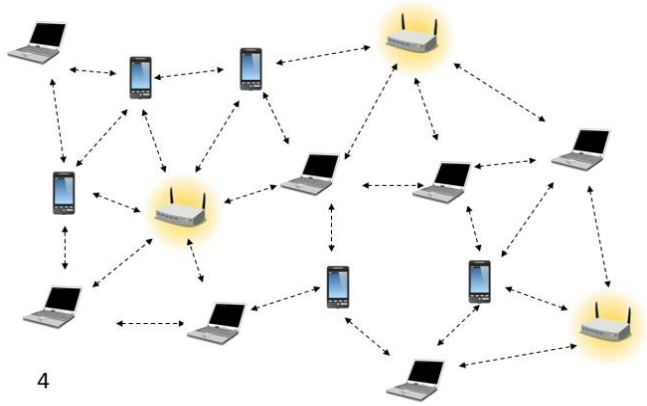

Fig -1: Connected devices through routers

Routing is performed for many kinds of networks, including the telephone network (circuit switching), electronic data networks (such as the Internet), and transportation networks. This article is concerned primarily with routing in electronic data networks using packet switching technology. In packet switching networks, routing directs packet forwarding (the transit of logically addressed network packets from their source toward their ultimate destination) through intermediate nodes. Intermediate nodes are typically network hardware devices such as routers, bridges, gateways, firewalls, or switches. General purpose computers can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables, which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time. Multipath routing techniques enable the use of multiple alternative paths. In case of overlapping/equal routes, algorithms consider the following elements to decide which routes to install into the routing table (sorted by priority):

Prefix-Length: where longer subnet masks are preferred (independent of whether it is within a routing protocol or over different routing protocol)

Metric: where a lower metric/cost is preferred (only valid within one and the same routing protocol)

Administrative distance: where a route learned from a more reliable routing protocol is preferred (only valid between different routing protocols)

2. RESEARCH ELABORATION

2.1 Algorithmic strategies used in routing

1. Brute force algorithm
2. Greedy strategy
3. Dynamic programming
4. Backtracking
5. Branch and Bound
6. Divide and Conquer
7. Decrease and Conquer
8. Transfer and Conquer

2.2 Types of Routing Algorithms

1. Link state routing:

Link-state routing protocols are one of the two main classes of routing protocols used in packet switching networks for computer communications, the other being distance-vector routing protocols. Examples of link-state routing protocols include open shortest path first (OSPF) and intermediate system to intermediate system (IS-IS).The link-state protocol is performed by every switching node in the network (i.e., nodes that are prepared to forward packets; in the Internet, these are called routers). The basic concept of link-state routing is that every node constructs a map of the connectivity to the network, in the form of a graph, showing which nodes are connected to which other nodes. Each node then independently calculates the next best logical path from it to every possible destination in the network. The collection of best paths will then form the node's routing table. This contrasts with distance-vector routing protocols, which work by having each node share its routing table with its neighbours. In a link-state protocol the only information passed between nodes is connectivity related. Strategy used : greedy programming, generally a variant of Dijkstra's algorithm is used.

2. Distance vector routing:

In computer communication theory relating to packet-switched networks, a distance-vector routing protocol is one of the two major classes of intra domain routing protocols, the other major class being the link-state protocol.

Distance-vector routing protocols use the Bellman–Ford algorithm, Ford–Fulkerson algorithm, or DUAL FSM (in the case of Cisco Systems's protocols) to calculate paths. A distance-vector routing protocol requires that a router inform its neighbors of topology changes periodically. Compared to link-state protocols, which require a router to inform all the nodes in a network of topology changes, distance-vector routing protocols have less computational complexity and message overhead. The term distance vector refers to the fact that the protocol manipulates vectors (arrays) of distances to other nodes in the network. The vector distance algorithm was the original ARPANET routing algorithm and was also used in the internet under the name of RIP (Routing Information Protocol). Examples of distance-vector routing protocols include RIPv1 and IGRP.

Strategy used : dynamic programming, generally bellman ford algorithm.

are CLI driven. The network model/configuration describes the state of the network (nodes, routers, switches, links) and the events (data transmissions, packet error etc.). An important output of simulations are the trace files. Trace files log every packet, every event that occurred in the simulation and are used for analysis. Network simulators can also provide other tools to facilitate visual analysis of trends and potential trouble spots.

Simulation of networks is a very complex task. For example, if congestion is high, then estimation of the average occupancy is challenging because of high variance. To estimate the likelihood of a buffer overflow in a network, the time required for an accurate answer can be extremely large. Specialized techniques such as "control variates" and "importance sampling" have been developed to speed simulation.

The network simulator must enable a user to:

1. Model the network topology specifying the nodes on the network and the links between those nodes
2. Model the application flow (traffic) between the nodes
3. Providing network performance metrics as output
4. Visualization of the packet flow
5. Logging of packet / events for drill down analyses or debugging

The "ns-3" simulation software is built using C++ and Python with scripting capability. The ns-3 library is wrapped by Python thanks to the pybindgen library which delegates the parsing of the ns-3 C++ headers to gccxml and pygccxml to automatically generate the corresponding C++ binding glue. These automatically-generated C++ files are finally compiled into the ns-3 Python module to allow users to interact with the C++ ns-3 models and core through Python scripts. The ns-3 simulator features an integrated attribute-based system to manage default and per-instance values for simulation parameters. All of the configurable default values for parameters are managed by this system, integrated with command-line argument processing. The large majority of its users focuses on wireless simulations which involve models for Wi-Fi.

3.3 Network topology

The general process of creating a simulation can be divided into several steps:

1. Topology definition: to ease the creation of basic facilities and define their interrelationships, ns-3 has a system of containers and helpers that facilitates this process.
2. Model development: models are added to simulation (for example, UDP, IPv4, point-to-point devices and links, applications); most of the time this is done using helpers.
3. Node and link configuration: models set their default values (for example, the size of packets sent by an application or MTU of a point-to-point link); most of the time this is done using the attribute system.
4. Execution: simulation facilities generate events, data requested by the user is logged.
5. Performance analysis: after the simulation is finished and data is available as a time-stamped event trace. This data can then be statistically analysed with tools like R to draw conclusions.
6. Graphical Visualization: raw or processed data collected in a simulation can be graphed using tools like Gnuplot, matplotlib or XGRAPH.

The selection of a network topology can affect:

1. Type of equipment the network needs.
2. Capabilities of the equipment.
3. Growth of the network.
4. Way the network is managed.

Standard Topologies:

1. Bus – Devices connected to a common, shared cable.
2. Star - Connecting computers to cable segments branch out from a single point, or hub.
3. Ring - Connecting computers to cable that form a loop.
4. Mesh – Connects all computers in a network to each other with separate cables.

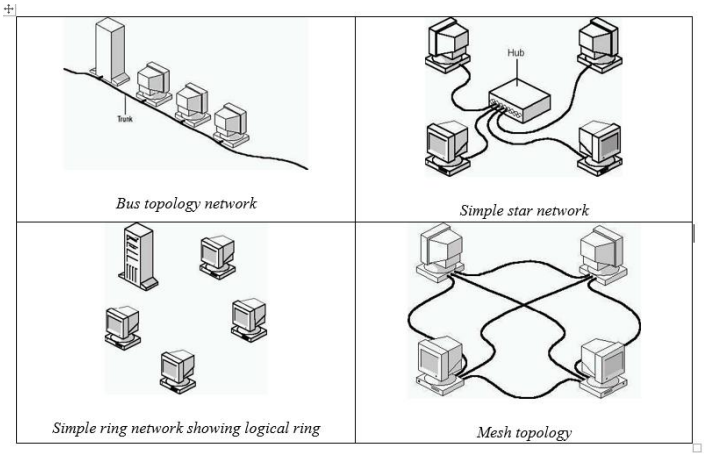

Fig-3: Common network topologies

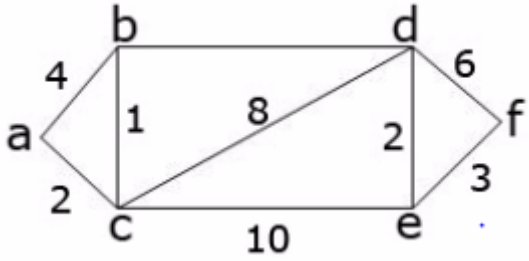

Fig-4: Network topology used