CS-358

End Semester Assignment

Name : P.V. SRIRAM
Roll No : 1801CS37

18ACS37

1)

a) Rule: | IP_Src = 128.119/16, IP_Dst = 128.121/16, Src_Port = Any, IP_Protocol = TCP, Action = Forward (4) |

Unless the router needs to block packets from exiting based on the content, Dst_Port will be **Any**

b) If the router is regulating content entering or exiting, this will be the protocol associated with the content or 'any'

IP_Protocol = **TCP & UDP** (S1→S4→S3, S1→S2→S3)

c) Rule: |IP_Src = 128.119/16, IP_Dst = 128.121/16, Src_Port = Any, Dst_Port = Any, IP_Proto = UDP, Action = Forward (2)|

The source will be the subnet of where the packet originated **IP_Src = 128.119/16**

d) Since the default behaviour of a router is the drop an unmatched packet, the action will be **Forward**

e) The interface will be the one attached to the next router in the packets path. **3**

2)

a) After the algorithm converges, we get the following distance vector for X'

$\{11, 8, 9, 0, 5\}$ → Represents shortest distance from X

b) Initial distance vectors for route 'Y' is $\{\infty, \infty, \infty, 5, 0\}$

c) Count to infinity problem

3)

a) Shortest distance for 'X' to 'V' is 3

So, for link X, the cost associated with this link is 3

$\therefore X' = 3$

b) Given shortest path from X to Z = 5, which is along the path $X \rightarrow W \rightarrow Z$

So, there is no way we can find the link 'Y'

N/A

4) Given

LAN consists of 10 computers connected by two self-learning Ethernet switches.

Switch Table entries are provided, Based on the tables.

a)

At t = 4, the node communicating is C

For TTL = 4, only one entry is there in table on left corresponding to C.

b) At t = 8, nodes C, F are communication

For TTL = 8, we have 2 entries of C and F in table on the right.

c) At t = 3, node J is communicating, because at TTL = 3, there is an entry for J in each table

d) At t = 2, node H is communicating

TTL = 2, we have an entry for H in each table

5) Given,

LAN with 10 computers connected by 2 self-learning switches.

Also given, signals sent are,

$t = 1 \Rightarrow F \rightarrow I$

$t = 2 \Rightarrow C \rightarrow J$

$t = 3 \Rightarrow E \rightarrow G$

$t = 4 \Rightarrow F \rightarrow C$

After filling switch Table,

MAC Addr	Interface	TTL
F	8	1
I	12	1
C	8	2
J	13	2
E	8	3
G	10	3

MAC	Interface	TTL
F	6	1
I	7	1
C	3	2
J	7	2
E	5	3
G	7	3

a) At $t=1$, source for switch 1 is (F, 6)

b) At $t=1$, destination entry for switch 1 is (I, 7)

c) At $t=2$, destination entry for switch 2 is (J, 13)

d) At $t=3$, source entry for switch 1 is (E, 5)

e) At $t=3$, destination entry for switch 2 is (G, 10)

6)

Given,

A CRC Algorithm with Generator (G) as 1001.

Data (D) = 10011010

$r = 3$

Required to find CRC bits for given data

⇒ We need to get the remainder when $D \cdot 2^r$ is divided by G

$D \cdot 2^r$ = D followed by r zeros = 10011010000

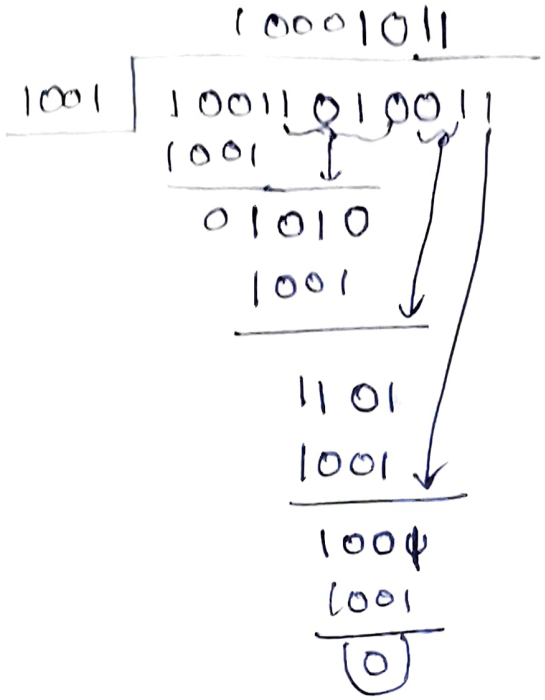
```
          1   0001  011
      ┌──────────────────
1001  │ 1001  1010  000
        1001
        ----
         0101 0
          1001
          ----
           1100
           1001
           ----
            1010
            1001
            ----
             0011
```

⇒ Remainder = 011

⇒ R = 011

⇒ Data to be sent is 10011010011

Verification



⇒ Remainder is zero.

∴ No loss / Corruption of Data

R = 011 is the correct CRC bits

7)

From the given figure, let us consider a packet/frame Travelling from D to A.

a) At point 5, the data's source is D

⇒ 1A-79-BC-E9-E6-3A

b) The packet is going to router

⇒ Destination MAC = D0-AE-4D-72-CC-F0

c) As source is D

⇒ Source IP is 128.119.58.43

d) As destination is A

⇒ Destination IP = 128.119.190.195

e) At point 2, Source is Router

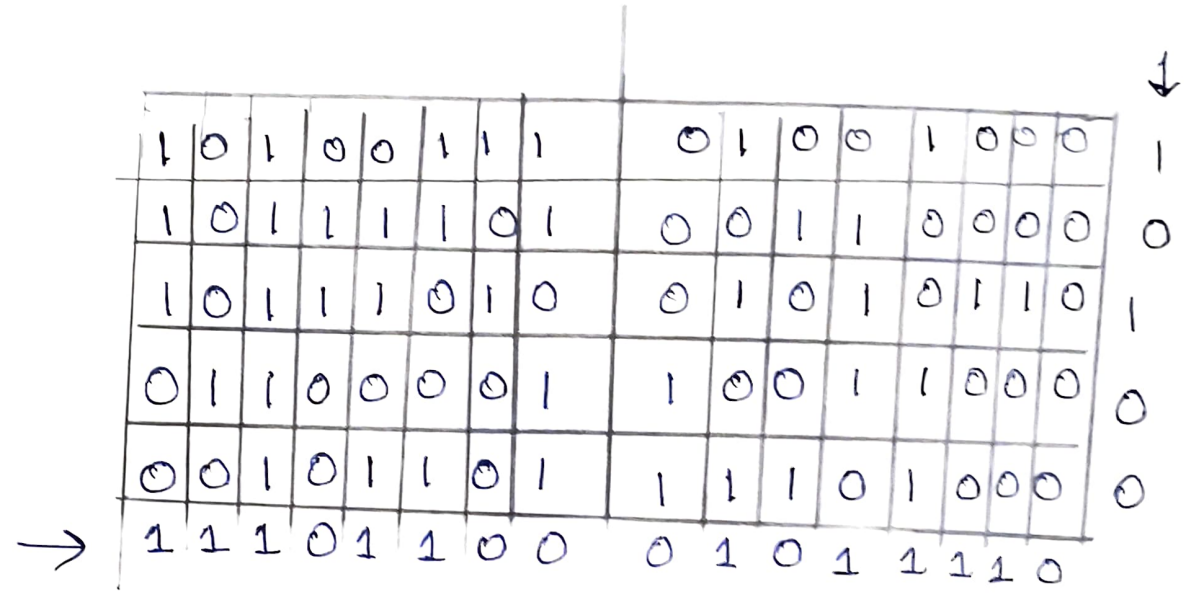
⇒ Source MAC = E6-E3-64-04-2A-D3

f) Destination is A

⇒ Destination MAC = C2-32-12-7D-4E-86

8) Given,

Payload consists of 10 eight bit values, given in the matrix below



	1	0	1	0	0	1	1	1	0	1	0	0	1	0	0	0	↓ 1
	1	0	1	1	1	1	0	1	0	0	1	1	0	0	0	0	0
	1	0	1	1	1	0	1	0	0	1	0	1	0	1	1	0	1
	0	1	1	0	0	0	0	1	1	0	0	1	1	0	0	0	0
	0	0	1	0	1	1	0	1	1	1	1	0	1	0	0	0	0
→	1	1	1	0	1	1	0	0	0	1	0	1	1	1	1	0	

Parity will be determined by no. of 1's in a row / column

i.e) Even ⇒ 0 / Odd ⇒ 1

As per the above rule,

a) Two-dimensional parity for 16 columns ⇒

1 1 1 0 1 1 0 0 0 1 0 1 1 1 1 0

b) 2-dimensional parity for 5 rows ⇒

1 0 1 0 0

9) Oscillation problem faced in the traditional computer network :

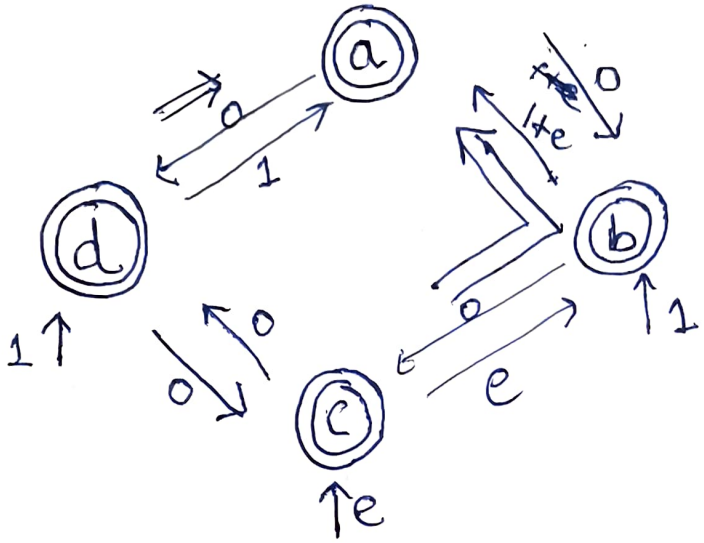
Let us consider example of Dijkstra's Algorithm where oscillation could be possible.

Link costs between any two nodes in the network are dependent upon the amount of traffic volume passing through the link at any specific time. ('dynamic cost policy)

Sample Scenario for example :-

(i) Four nodes : a, b, c, d
(ii) Link cost is equal to amount of load carried on the link.
(iii) Nodes b, c, d each route traffic only to node 'a'
(iv) 'b' and 'd' send 'd' unit amount of load to 'a'
(v) 'c' send 'e' amount of load to 'a' ($e < 1$)

Initial routing:



(Double arrows represent path)

Fig 1

From above network topology, we can see that at this specific instant

(i) Cost for clockwise path from C to A = 1

(ii) cost for counterclockwise path from C to A = 1 + 2e

So, C's least cost path to a is clockwise, so we have to change our initial route.

Similarly b's least cost path to a is also clockwise (1+e)
rather than currently selected counterclockwise

Modified Routing:

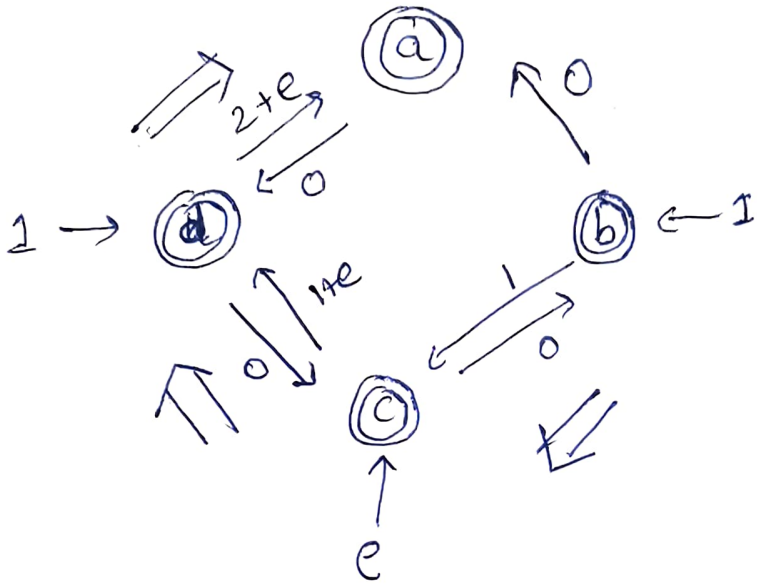


Fig 2

Now, on the next run of the algorithm, all the nodes b, c and d detect a zero cost path to 'a' in the counterclockwise direction and corresponding the traffic is routed along minimum cost direction

Modified Routing:

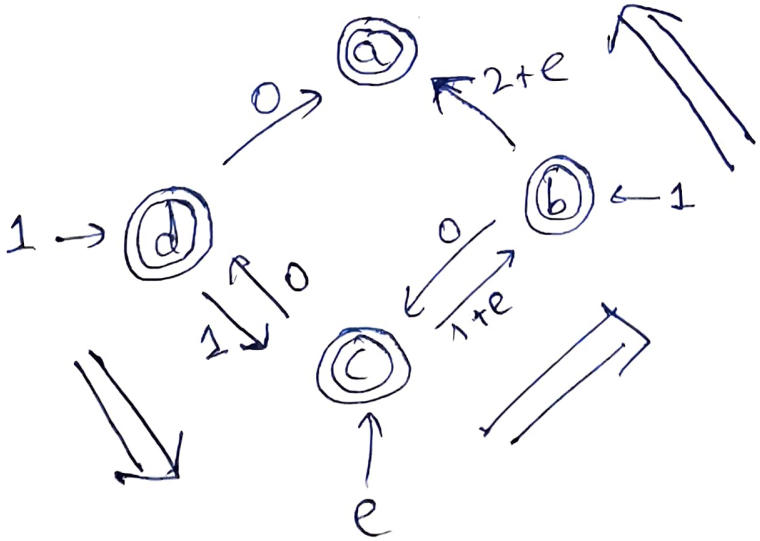


Fig 3

Again on the next run, nodes b, c, d detect 0 cost path to 'a' in the clockwise direction and traffic is again routed exactly similar to Fig 2. These modifications will oscillate from clockwise to counterclockwise or vice-versa as long as routing is needed to be done.

This exact scenario is known as oscillation problem in traditional computer network. Due to this many routing algorithms do not define link costs based on short term load levels. Software Defined Networks has been a major advancement which can be used to overcome oscillation problem.

10)

a) Motivation for adopting SDN over traditional Network

a) Complex Routers:

Routers consist of two major parts software and Hardware. Routing software requires millions of lines of source code, which is hard to extend. While hardware consist of tens of billions of transistors using lot of power and memory at the same time. We have to integrate many compler functions like OSPF, BGP, multicast, QoS, Traffic engineering, NAT, firewalls, MPLS etc.

b) Oscillation Problem:

In this dynamic link costs lead to oscillating route behaviour. This leads to modification of forwarding table numerous times leading to larger overhead.

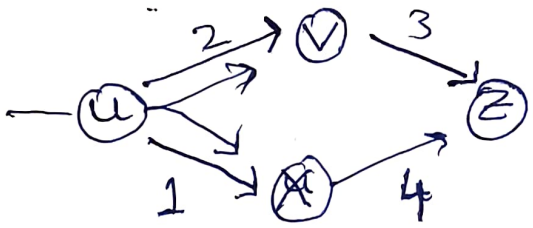
c) Traffic Engineering

Too difficult in traditional network

Ccs

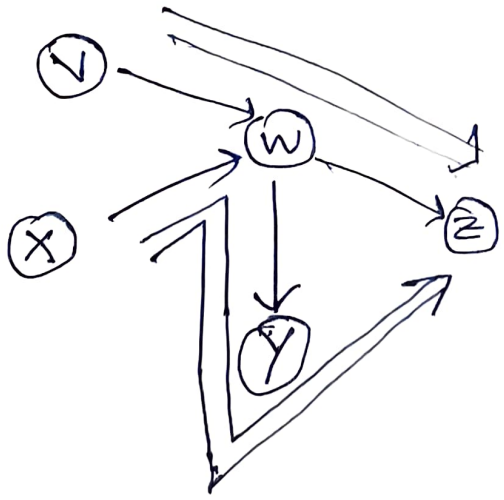
Case I : load balancing

If network operator wants to split u-to-z traffic along uvz and uxz, then he/she has to use new routing algorithm, not possible to do it



Case II : Differentiation on basis of source of load

If we want to route traffic from x and that from v coming into w differently to z, it is not possible within existing setup (As it uses destination based forwarding, link state and distance vector routing



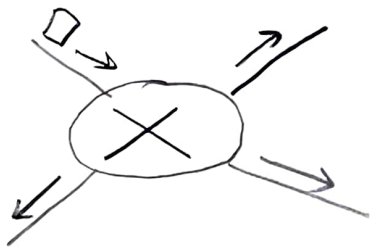
b) The concept of networks substrate comes when we talk about SDN. It is used for being able to mimic the success in the software industry.

Characteristics include :

(i) it has simple common substrate

(ii) Building OS on top of the hardware, which enables easy deployment of networking applications.

Router Example



Basic job of a router is receiving packets, checking the routing table, forwarding the packets out. For building routing table, the router has to understand BGP, OSPF, RIP etc. We can use network substrates to get the routing table from somewhere else (centralized)

This can also be seen in the separate data and control plane based architecture of SDN. Since control plane is centralized and sits on top of all the software and hardware, hence it is networks substrate.

11) Software Defined Networking (SDN)

Is the physical seperation of the network control plane from the forwarding plane and where a control plane controls several devices. SDN makes networks agile and flexible. It provides better network control for the cloud computing service providers to respond quickly to ever changing business requirements. Hence it is preffered over traditional network architecture

A typical representation of SDN architecture includes three layers.

(i) Application Layer

(ii) Control Layer

(iii) Infrastructure Layer

SDN architecture Diagram

Network Control Applications

Application Layer

routing

Load Balance

Access Control

SDN Northbound interface

A-CPIs Application controller plane interface

Control Layer

Interface, abstractions for network control

Network Graph

RESTful API

. . . . .

Intent

Statistics

. . . . .

Flow Tables

Network-wide distributed robust state management

Link-state info

Host info

. . . . . .

switch info

OpenFlow

. . . . .

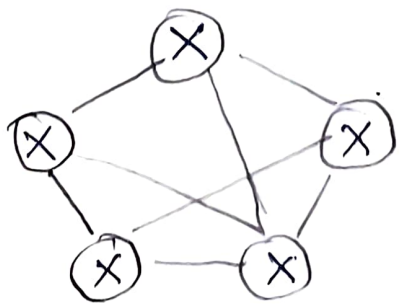
SNMP

Communication to/from controlled devices

D-CPI : Data controller plane interface

SDN Southbound Interface

Data Layer



SDN-controlled switches

SDN Architecture

Data Plane

The data plane consists of data sources and sinks. It functions as a Traffic forwarding / processing engine and may have the ability to handle protocols like ARP, LLDP. It has interfaces communicating to the control plane.

Functions:

(i) Programmatic Control of all functions offered by network element.

(ii) Capability advertisement

(iii) Event notification

Control Plane

The control layer is considered as the brain of SDN. It communicates with the data plane using south-bound APIs. The intelligence to this layer is provided by centralized SDN controller software. This controller resides on a server and manages policies and the flow of traffic throughout the network. The physical switches in the network constitute the infrastructure layer. It interfaces to the application plane.

Functions:

(i) Topology and network state information

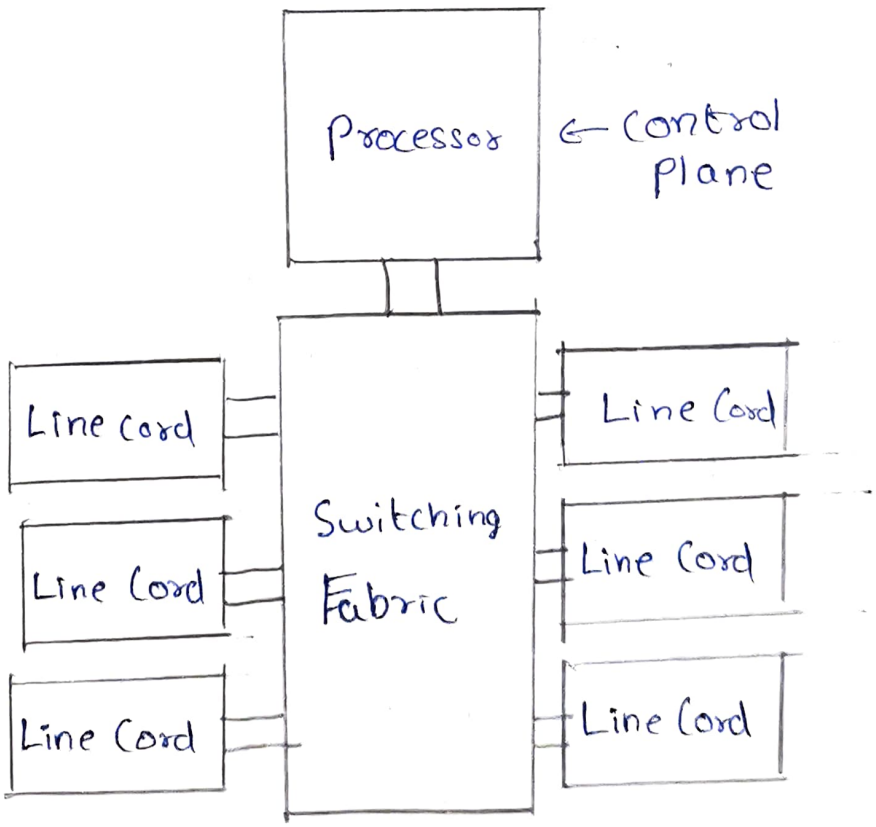
(ii) Device Discovery

(iii) Path Computation

(iv) Security mechanism

Application Plane

The SDN application layer contains network applications or functions such as intrusion detection systems, load balancing or firewalls. It communicates with the control layer using the northbound API



Interaction between Data and Control Planes via Switching Fabric

12) Data Center transport impairments and requirements

The shallow packet buffers cause 3 specific performance impairments.

(I) Incast

If many flows converge on same interface of a switch over short period of time, packets may exhaust either switch memory or maximum permitted buffer resulting in data losses. This traffic pattern arises naturally from use of partition/aggregate pattern as request for data creates incast at queue of switch port connected to aggregator.

We capture incast instances via packet level monitoring. Since size of each individual response leads to loss of packets almost invariably results in a TCP timeout. Thus, whenever time out occurs response almost always misses aggregator's deadline.

(II) Queue buildup

Long lived, greedy TCP flows will cause length of bottleneck queue to grow until packets are dropped resulting in familiar saw tooth patterns.

Two impairment occurs when short and long flows transverse same queue. Even when no packets are lost, short flows experience increased latency as they are in queue behind packets from large flows, thus queue build up impairements occurs and this traffic pattern occurs frequently.

(III) Buffer Pressure

It is very common for short flows on one part to be impacted by activity on any of many ports.

Indeed, the loss rate of short flows in this traffic pattern depends on no. of long flows traversing other ports. The long, greed TCP flows build up queues on their interfaces. Since buffer space is a shared resoure, the queue build up reduces the amount of buffer space available to absorb bursts of traffic from partition / Aggregate Traffic. This is Buffer pressure.

Tension between requirements

High Throughput High Burst tolerance	Low Latency
Deep buffers Reduced RTO_{min} → Doesn't help latency	Shallow buffers AQM-RED → Avg queue not fast enough for incast

Case Study

Microsoft Bing

(i) Measurements from 6000 server production cluster

(ii) Instrumentation passively collects logs
   (a) Application level
   (b) Socket level
   (c) Selected packet level

(iii) More than 150 TB of compressed data over a month.

Whereas in the case of Facebook, it uses Partition/Aggregate

Partition/Aggregate: Foundation for many large scale web apps.

Facebook

(i) Uses partition/Aggregate ⇒ Multiget

Aggregators : Web servers

Workers : Memcached Servers

Workloads are:

(i) Partition / Aggregate → Delay Sensitive

(ii) Short messages (50KB, 1MB) → Delay Sensitive

(iii) Large Flows (1MB-50MB) → Throughput Sensitive

13)

a) Forwarding Abstraction

Purpose: Abstract away forwarding hardware while still be able to express how and where to forward a packet.

The first abstraction relates to how network elements forward packets. At a high enough level of abstraction all network elements perform the same task.

Abstraction 1: Packet forwarding as a computational problem.

The function of any network element (NE) is to

(i) Receive a packet
(ii) Observe packet fields
(iii) Apply algorithms
(iv) Optionally edit the packet
(v) Forward or discard the packet.

Features:

⇒ Flexible

(i) Forwarding behaviour specified by control plane
(ii) Built from basic set of forwarding primitives

=> Minimal

(i) Streamlined for speed and low power

(ii) Open and not vendor specific.

Openflow is an example of forwarding abstraction

Openflow Forwarding Abstraction

=> Protocol independent

(i) Construct ethernet, IPV_4, VLAN, MPLS

(ii) Construct new forwarding methods

=> Backward compatible

(i) Runs in existing networks

=> Technology independent

(i) Switches, routers, WiFi APs

(ii) Cellular base stations

(iii) WDM / TDM circuits

b)

Network Functions Visualization

It is a way to virtualize network services, such as routers, firewalls and load balancers that have traditionally been run on proprietary hardware. These, services are packaged as virtual machines on commodity hardware, which allows service providers to run their network on the standard servers instead of proprietary ones. It is one of the primary components of a 'telco' cloud, which is reshaping the telecommu -nication industry.

With NFV we dont need to have dedicated hardware for each network function. NVF improves scalability and agility by allowing service providers to deliver new network services and applications on demand, without requiring additional hardware resources.

NVF architecture:

(i) Virtualized network Functions:
These are software applications that deliver network functions such as file sharing, directory services and IP configuration

(ii) Network function virtualization infrastructure

It consist of the infrastructure components - compute, storage, networking.

(iii) Management, automation and network orchestration

It provides a framework for managing NFV and VNF.

Benefits:

With NFV service providers can run network function on standard hardware, instead of dedicated hardware.

NVF gives providers the flexibility to run VNFs accross different servers or move them around as needed when demand changes. This flexibility lets service providers delivers services and apps faster

(C) Service Function chaining

It is a capability that uses SDN capabilities to create a chain of connected network services, such as L4-7 services like firewalls network address translation (NAT), and intrusion protection.

Network operators can use this to setup suites or catalogs of pathways for traffic to travel through. Any one path can consist of any combination of connected services depending on the traffics requirements. Different traffic requirements might be more securely lower latency, or an overall high quality of service (QoS)

The primary advantage of network service chaining is to automate the way virtual network connections can be setup to handle traffic flows for connected services.

The chaining capability automates what traditional network administrators do when they connect up a series of physical L4-7 devices to process incoming and outcoming network traffic.

Another benefit is when used in combination with SDN, is optimising the use of network resources and improving application performance, SDN analytics and performance tools can use the best available network resources and help negotiate around network congestion issues.

Service chaining is facing a new networking technology that is purposefully trying to get away from service chaining. It is called SASE.

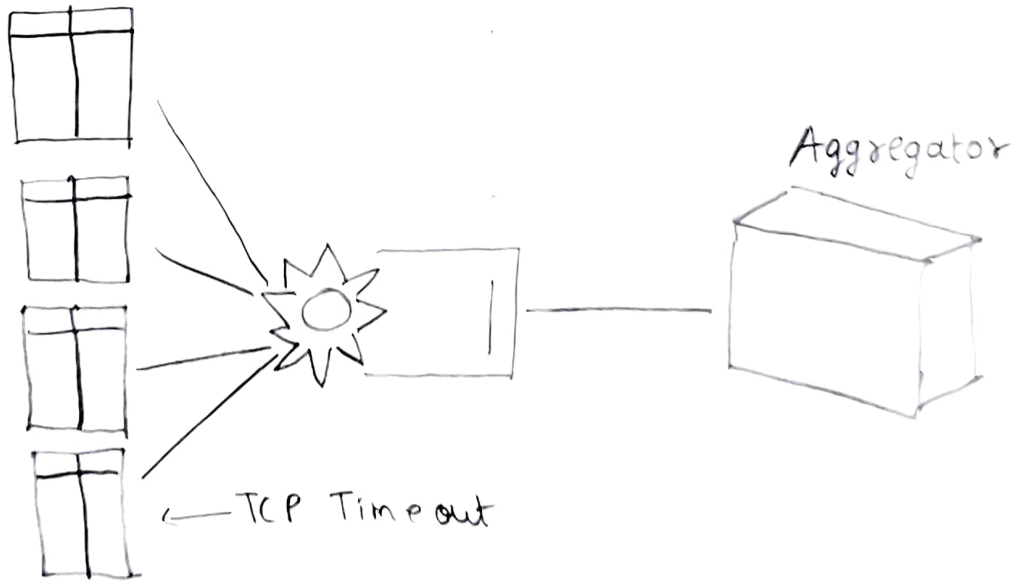
The technology's capabilities mean that a large number of virtual network functions can be connected together in a NFV environment. These connections can be set up and torn down as needed with services chain provisioning through the NFV.

14) TCP Incast problem

TCP incast problem refers to TCP performance degradation when it runs in many to one traffic pattern.

Initially catastrophic collapse is observed when more no. of senders concurrently transmit data blocks to a single receiver and any sender is not allowed to send the next block until all senders finish transmitting.

Therefore, in the delay-sensitive online application which employ partition / aggretion computing very few senders extended the time, so the system responses are excluded from the final result due to the missing aggregator deadlines.



Solutions to solve TCP incast Problem

1) Adjusting system parameters

It improves TCP incast problem by adjusting parameters, such as changing block size, enlarging buffer size, and increasing capacity.

Limiting the no. of concurrent flows is also a way to improve situation

2) Designing Enhanced Mechanisms

(i) Reduce RTO_{min} to reduce the throughput collapse

(ii) Shrinking MTU (Max Transmission unit)

3) Replacing loss-based TCP version

(i) Adjusting congestion window properly according to the delay information generated by RTT measurement

Slight Adjustment of window size is beneficial to protect the buffer.

4) New Transmission protocols.

DCTCP works because

(a) High Burst Tolerance
* Large Buffer Headroom → bursts fit
* Aggresive marking → sources react before packets are dropped

(b) Low latency
Small buffer occupancies → Low queuing delay

(c) High throughput
ECN averaging → Smooth rate adjustments, low variance

15) Markov - Decision Problem (MDP)

Markov Decision Processes are discrete-time stochastic control processes. They provide a mathematical framework for modelling decision making in situations where outcomes are partly random and partly under the control of a decision maker. Usually used for studying Optimization problems utilizing Dynamic Programming.

Problem :

MDP's are modelled as follows,

* 4-tuple (S, A, P_a, R_a) where:

(i) S is a set of states called state space

(ii) A is a set of actions called the action space

(iii) $P_a(s, s') = Pr(S_{t+1} = s' \mid S_t = s, a_t = a)$

(Probability that action a in state s at time t will lead to state s' at time t+1)

(iv) $R_a(s, s')$ is the immediate reward (Expected) received after transitioning from state s to s', due to action a.

NOTE: The state and action spaces may be finite / infinite.

A policy function π is a probabilistic mapping from state space to action space.

Objective:

The objective of an MDP process is simply to find the optimal policy ($\pi(s)$) mapping from state space (S) to action (A) space.

To do this, we use a function called value function,

$$V(s) = \sum_{s'} P_{\pi(s)}(s, s') \left(R_{\pi(s)}(s, s') + \gamma V(s') \right)$$

This value function would return the expected returns of all Rewards possible from state s.

And naturally, we would want a policy function which would maximise the expected return.

$$\pi^*(s) = \arg\max_a \left\{ \sum_{s'} P(s' \mid s, a) \left[R(s' \mid s, a) + \gamma V(s') \right] \right\}$$

We can converge to this optimal policy by iterating through 2-steps.

(i) Value Update

(ii) Policy Update

which are repeated in some order for all states until no further change takes place. Both recursively update a new estimation of the optimal policy and state value using an older estimation of those values

Example

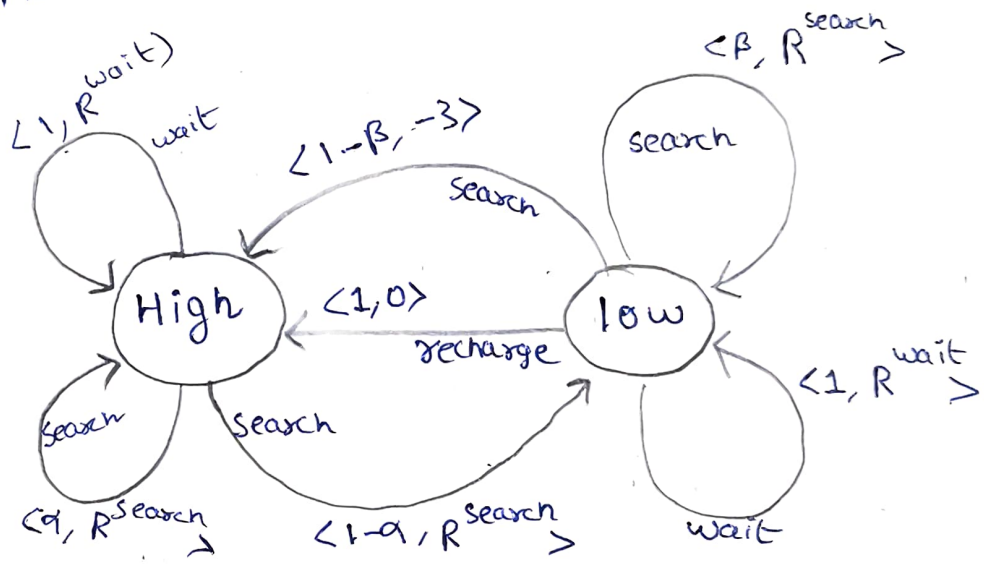
Consider, A robot whose purpose is to pick up litter.

2 states $S = \{ high, low \}$ ← Battery Charge

Actions $A = \{ wait, search, Recharge \}$

- wait: Hold still
- search: Search for litter
- Recharge: Go to charging Station

The MDP could be modelled as,



Note: Edges are represented by $\langle P, R \rangle$

- P: Probability of action
- R: Reward

Q-Learning

Q-Learning is a model-free RL algorithm to learn the value of an action in a particular state. It does not require a model of the environment and it can handle problems with stochastic transitions and rewards without requiring adaptations.

For any finite MDP (FMDP), Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over any and all successive steps, starting from the current state. Q-learning can identify an optimal action-selection policy for any given FMDP, given infinite exploration time and a partly-random policy.

Note: "Q" refers to the function that the algorithm computes - the expected rewards for an action taken in a given state.

Objective

Through this algorithm, we estimate the optimal Q-function.

We assume a Q table, mapping state action pairs to values

i.e) $Q : S \times A \rightarrow R$

Before learning begins, Q is initialized to a possibly arbitrary values. Then at each time 't' the agent selects an action a_t, observes a reward r_t, enters a new state s_{t+1} and Q is updated.

$$Q'(s_t, a_t) = Q(s_t, a_t) + \alpha \left(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right)$$

- $Q'(s_t, a_t)$: new value
- $Q(s_t, a_t)$: old value
- α: learning rate
- r_t: reward
- γ: discount factor
- $\max_a Q(s_{t+1}, a)$: optimal future value (Expected)
- $Q(s_t, a_t)$: old value

$r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$: Temporal Difference

Three factors,

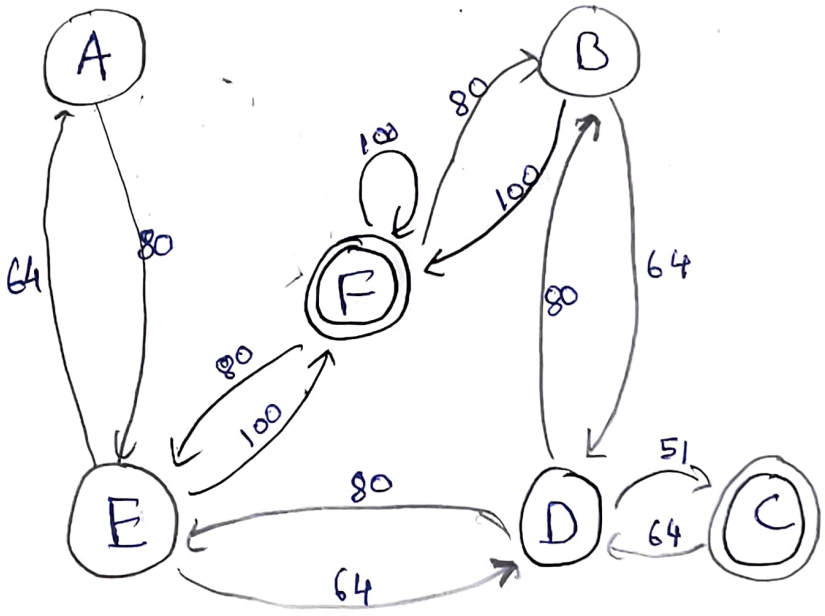
(i) $(1-\alpha) Q(s_t, a_t)$: The current value weighted by learning rate.

(ii) αr_t : Reward if action a_t is taken when in s_t.

(iii) $\alpha \gamma \max_a Q(s_{t+1}, a)$: Maximum reward that could be obtained from s_{t+1}

Example

Consider the following environment



Initial Q-Table,

$$Q = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Final Q-Table,

$$Q = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} - & - & - & - & 80 & - \\ - & - & - & 64 & - & 100 \\ - & - & - & 64 & - & - \\ - & 80 & 51 & - & 80 & - \\ 64 & - & - & 64 & - & 100 \\ - & 80 & - & - & 80 & 100 \end{bmatrix} \end{array}$$

Markov decision Process	Q-Learning
(i) Discrete Time stochastic Control process	(i) Continuous Time stochastic Control process
(ii) Non-Temporal Difference problem	(ii) Temporal Difference problem
(iii) Model based / Policy based algorithm	(iii) Model free algorithm
(iv) Utilises Dynamic programming	(iv) Utilises Bellman equation
(v) Uses value function and policy function for optimization	(v) Utilises Q-Table for optimisation
(vi) Used when R, P are known prehand	(vi) Used when R, P are unpredictable

* MDP Performs better than Q-Learning in non-Temporal environments i.e) When the episodes end at a time stamp.
* MDP does not suffer from bias as each update is made using a true sample of what Q(s, a) should be.

16) Exploration and Exploitation in Q-Learning

The Q-learning algorithm does not specify what the agent should actually do. The agent learns a Q-function that can be used to determine an optimal action. There are two things that are useful for the agent to do:

Exploit: The knowledge that it has found for the current state s by doing one of the actions 'a' that maximises Q[s, a]

Explore: In order to build a better estimate of the optimal Q-function. i.e) It should select a different action from the one that it currently thinks is best

Tradeoff

During learning, it might be the case that the agent chooses the action value which has highest estimate (greedy action). Or something else (non-greedy action). Doing only one of these doesn't help in convergence

Balancing rightly between exploitation and exploration is the key. Only exploitation might be good in short run but might miss the long run optimum. Therefore, even though initially exploration gives bad results it will help in convergence.

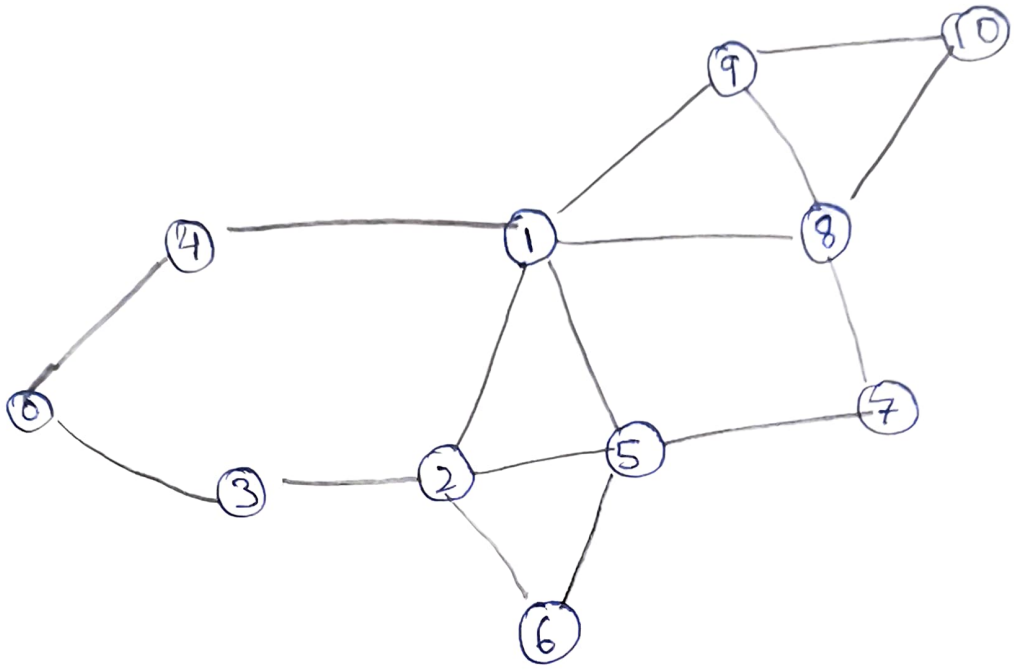
Some algorithms that balance these both are

(i) ϵ greedy algorithm
(ii) Upper confidence bound
(iii) Thompson sampling

Note: The property that needs to be held by exploration is "Degradation". Exploration must happen in the initial stages of learning and must die down eventually. Otherwise it would account to high variance.

Eg: Decaying ϵ greedy Policy

17) Given graph is,



Also given, optimal Q-matrix.

Now, we know that

$$\pi^*(s) = \underset{a}{\text{argmax}}\ (Q(s,a))$$

$\Rightarrow \pi^*(0) = 4$ (138 is max value in 0 row)

$\pi^*(4) = 1$ (174 " " " " 4 row)

$\pi^*(1) = 8/9$ (218 " " " " 1 row)

$\pi^*(8) = 10$ (274 " " " " 8 row)

$\pi^*(9) = 10$ (274 " " " " 9 row)

$\therefore$ Path $\Rightarrow$ $0 \to 4 \to 1 \to 8 \to 10$

(OR)

Path $\Rightarrow$ $0 \to 4 \to 1 \to 9 \to 10$