CS 358 ESE ASSIGNMENT

Navyasree
1801CS06
CS358

① Match-table entries:-

S_1 → match	Action
Ipsrc: 128.119/16 Ip.protocol: TCP	Forward (4)
Ipdst: 128.121/16	Forward (2)

S_2 → Match	Action
Ingress port=3 Ip Src=128.119/16 Ip dst = 128.121/16 Ip protocol = UDP	Forward (2)

S_3 — Match	action
ingress port=2 Ip.dst=128.121/16	Forward (1)
ingress port=4 Ip.dst=128.121/16	Fordward (1)

S_4 → Match	action
ingress port=2 ip.Src=128.119/16 ip.dst=128.121/16 Ip protocol=Tcp	Forward (3)

Navyasree
1801ES06
CS 358

(a) For router S1, Dst port Can be "any" Unless router needs to block packet from exting based on context, dst port will be 'any'

(b) For router S1, Both "TCP & UDP" are used. for interface 2, UDP used for path: S1 → S2 → S3
For interface 4, TCP used for path: S1 → S4 → S3

(c) For router S2, IP src will be "128.119/16" Source will be subnet of where packet originated

(d) Here, S3 forwards packets to 128.121/6 through interface 1. So Action is 'forward'

(e) For router S4, packets needs to be forwarded to interface "3" for it to reach S3.

Navyasree
1801CS06
CS358

(2) A1

(a) The distance Vector from router when algorithm converges, 'x', is

"{11, 8, 9, 10, 5}" (By finding Shortest distance from 'x')

(b) Initial distance Vectors for router 'y' is

$(u, v, w, x, y) = (\infty, \infty, \infty, 5, 0)$

(c) Name of problem which occurs when link costs increase is "Count to infinity problem"

(3) A1

(a) We know, Shortest distance for 'x' to 'v' is 3

So, for link x, the cost associated with the link is

$3 \Rightarrow \therefore x = 3$

(b) We know, Shortest path from x to z = 5.

which is along path $x \to w \to z$

There is no possibility we find link 'y'

∴ Hence, n/a

(4) A:

Given,

Two self learning switches (switches in which they learn which hosts can be through which interface)

Through table entries for both switches,

(a) At t=4, Nodes Communicating are C

In given table1, there is only packet transmission record to C at TTL=4

(b) At t=8, Nodes communicating are C, F

for TTL=8, packet transmission between C & F as recorded in table 2.

(c) At t=3, Nodes communicating are J

packet transmission through J is recorded both in table 1 & table 2 at packet whose TTL=3

(d) At t=2, Nodes Communicating are H

Packet transmission through H is recorded in both table 1 & table 2 whose TTL=2

Navyasree
1801CS06
CS358

(5) A:-

At t = 1 : F → 1

So packet transmissions recorded after this step,

MAC	interface	TTL
F	8	1
I	12	1

MAC	interface	TTL
F	6	1
I	7	1

At t = 2 : C → J

After C → J, table entries look like

MAC	interface	TTL
F	8	1
I	12	1
C	8	2
J	13	2

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

At t = 3 : E → G

After E → G, table entries look like.

MAC	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

Navya Sree
1801CS06
CS358

At t=4 : F→C

After these transimission from F→C

Final table looks,

MAC	Interface	TTL	MAC	Interface	TTL
F	8	1	F	6	1
I	12	1	I	7	1
C	8	2	C	3	2
J	13	2	J	7	2
E	8	3	E	5	3
G	10	3	G	7	3
F	8	4	F	6	4
C	8	4	C	3	4

(a) At t=1, we have seen in step1 Source entiies,

(F, 6) is added as an source entry to Switch 1

(b) A t = 1, in step1 we have seen that

(I, 7) is added as destination entry to Switch 1

(c) At t=2, we have seen in Step 2

(J, 13) is destination entry to Switch 2

Navya Sree
1801CS06
CS358

(d) At t=3, we have Seen in step 3

(E, 5) is Source entry for switch 1

(e) At t=3, we have Seen in step 3

(G, 10) is destination entry for Switch 2

(6) A:-

Cyclic Redundancy check algorithm (CRC):

CRC is used to detect accidental changes to data transmitted via networks. CRC involves binary division of data bits being sent by a pre-determined divisor agreed by communating System. Divisor is generated using co-efficient of polynomial. We append 'r' bits to datapayload(D) and store the remainder of (D) when divided by G and send the data (D + remainder).

Now,

We divide the data recieved on other side with the same divisor (G)

If remainder is '0' => No error in transmission

If remainder ≠ '0' => Error in transmission

Given, $r = 3$

Generator / Divisor (G) = 1001

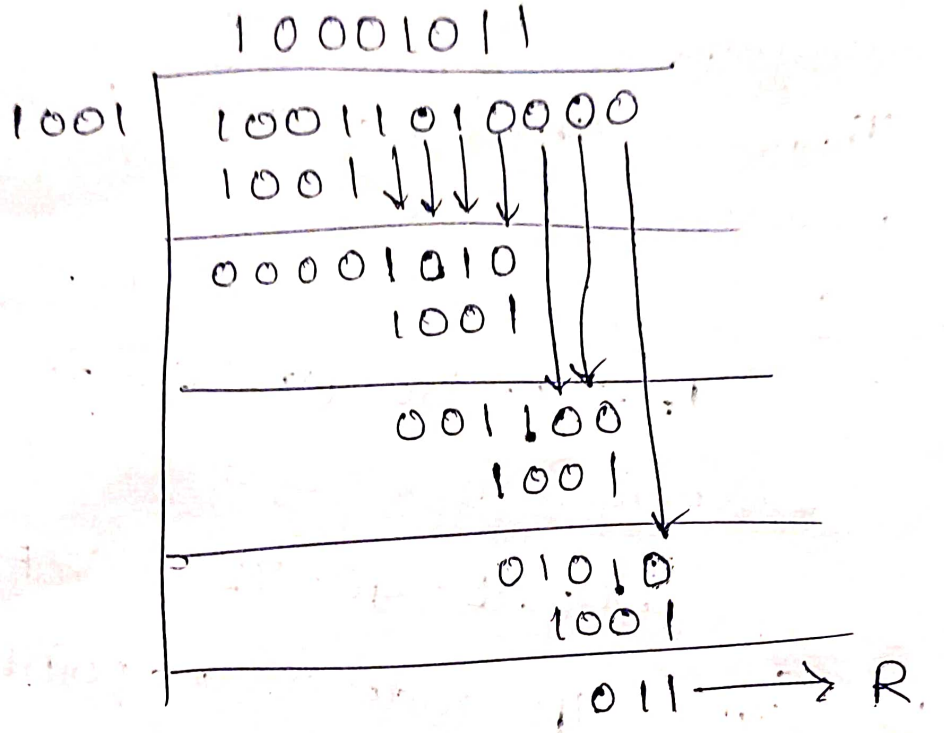
Datapayload (D) = 10011 0010

We add 'r' zeros to D, divide by G,

D+r = 10011010000 ('+' means appending)

G = 1001

on division,



We got that, Remainder (R) = 011

We will add 111 to D & Send the data

∴ CRC bits (R) associated with Datapayload (D) = 011

(7) Ans

Given,

Source and destination addresses of IP datagram & Ethernet address (MAC address) Ethernet is physical address, IP is logical address.

(a) Source mac address at point 5:

1A-79-BC-E9-E6-3A

(b) Destination mac address at point 5:

D0-AE-40-72-CC-F0

(c) Source IP address at point 5:

128.119.58.43

(d) Destination IP address at point 5:

128.119.190.195

(e) Source mac address at point 2:

E6-E3-64-04-2A-03

(f) Destination mac address at point 2:

C2-32-12-7D-4E-86

These are required mac & IP addresses at point 5, 2

Navya Sree
1801CS06
CS358

(5) A:

Given,

payload has 10 8-bit values.

Method:

calculate number of one's from top to bottom & left to right if the count is odd mark it as 1 else mark it as 0

1	0	1	0	0	1	1	1	0	1	0	0	1	0	0	0
1	0	1	1	1	1	0	1	0	0	1	1	0	0	0	0
1	0	1	1	1	0	1	0	0	1	0	1	0	1	1	0
0	1	1	0	0	0	0	1	1	0	0	1	1	0	0	0
0	0	1	0	1	1	0	1	1	1	1	0	1	0	0	0
⇒ 1	1	1	0	1	1	0	0	0	1	0	1	1	1	1	0

⇓ 1 0 1 0 0

(a) On Counting from top to bottom,

2-Dimensional parity for 16 columns → 1110110001011110

(b) On Counting from left to right,

2-Dimensional parity for 5 rows → 10100

(9) A:-

Let us now look at the Oscillation problem faced in traditional Computer network,.

We will Consider example dijkstra's algorithm where Oscillation occurs.

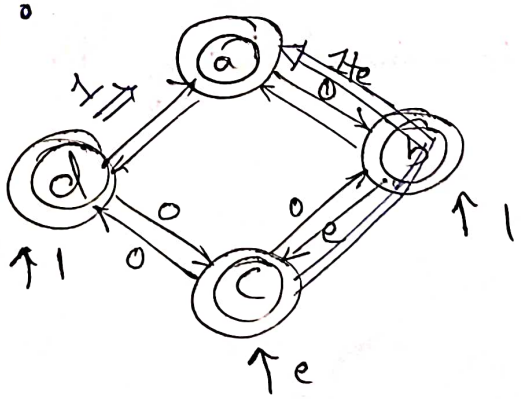
Link costs between any two nodes in the network are dependent on amount of traffic volume passing through link at any Specific time.

Example Case

There are 4 nodes a,b,c,d. Nodes b,c,d each route traffic only to node a. a Unit amount of load is Sent from b & d to a. e amount of load Sent from c to a.

> Link cost = amount of load Carried on link

Initial routing :-



(routes from Src to dest which are in use Currently marked by blue.)

Navyasree
1801cs06
CS358

We can Observe at this instant,

Costs of

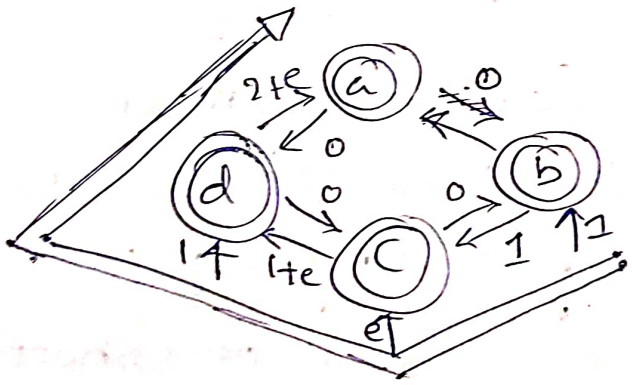
- clockwise path from c to a = 1
- Counter clockwise path from c to a = 1+2e

=> c's least cost path to a is Clockwise

There occurs a need to change our initial route.

In a similar way, b's least cost to a is also clockwise (1) rather than Currently Selected Counter clockwise (1+e)

Modified routing:

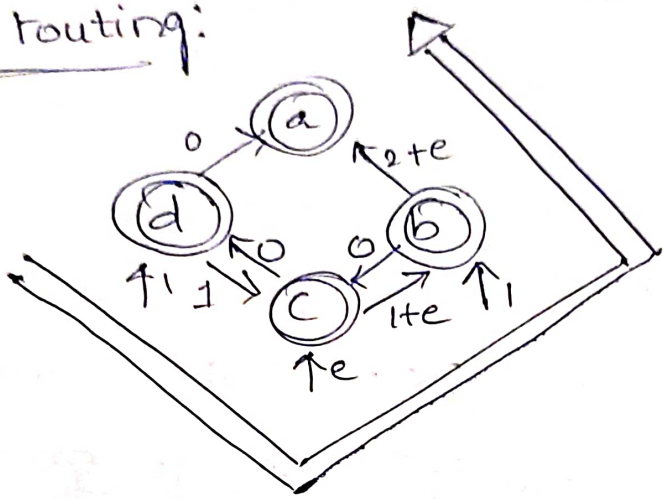


In the next run of algo, b,c,d detect zero cost path to a in anticlockwise direction & corresponding the traffic is routed along minimum cost direction

Navyasree
1801CS06
CS358

Now, on the next run

Modified routing:



Here nodes b, c, d detect 0 cost path to a in clockwise direction and traffic is again routed. Same as before fig. These modifications Oscillate. from clockwise to anti-clockwise / vice-versa as long as routing need 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 networking has been a major advancement which can be used to Overcome Oscillation problem.

Navyasree
1801CS06
CS358

(10) A:-

Motivation to adopt Software defined network Over traditional network

The reason is that the traditional networking is rooted in fixed-function network devices, for example a Switch or route. If the networks functions are implemented as hardware constructs then its speed is usually bolstered.

The reccuring hurdle for traditional networks is flexibility. Few APIs are exposed for provising Software, but this software can't be quickly modified as needed. Traditional networking consist of traits.

(i) Functions of traditional networking are primarily implemented from dedicated devices using one or more Switches, as well as routers and application delivery Controllers.

(ii) Functionality of traditional networking is largely implemented in dedicated hardware, Such as application-Specific

integrated circuits. One of the negative aspects of this traditional hardware-centric networking is its limitation.

SDN has advantage of generating the framework that bolsters data-intensive applications such as big data & virtualization.

In addition to these benefits here are some other reasons data enterprises network management, SDN offers following succinct advantages:

* Traffic programmability
* Greater agility
* Capacity to generate policy-driven network supervision
* Ability to implement network automation.

Here are the primary advantages that SDN offer:

* Centralized network provising.
* Holistic enterprise managem
* more granular security
* Lower Operating Cost and hardware savings & reduced Capital expenditure.
* Cloud abstraction
* Consistent & timely Content delivery.

In addition to these benefits here are some other reasons why data enterprises are choosing SDN

* SDN lets network operators and administrators adjust thier bandwidth as needed
* Traditional networks are rigid, making it difficult for network operators & administrators to customize the programming of their networks.

Traditional network architectures that are too old, rigid & expensive to scale. are out of alignment with today's hybrid cloud combination of traditional, public and private cloud and IT as a Service (ITaas) deployments. Gives great innovations & reductions in complexity.

For traditional network infrastructure, each switch determine where traffic goes and then directs the traffic based of these determinations. with SDN infrastructure, the process of determination and direction has been decoupled. Switches still direct the traffic how process of determining where the traffic goes is performed automated programmable interface called controller.

This platform allows network administrators to change network-wide settings with a centralized console. where as traditional network infrastructure might warrant deploying network changes in a piecemeal fashion for individual devices.

SDN infrastructure given network administrators the flexibility to change network traffic and enables network resources deployment that scales at the same speed as server and storage, redirecting it as needed.

Additionally, the SDN Controller reduces complexity and enables the network to scale as needed

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

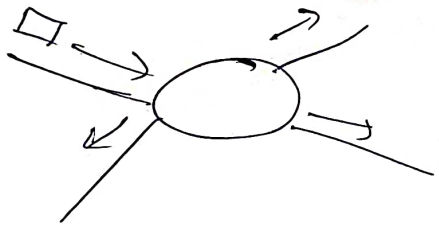
Navya Sree
1BO1CS06
CS358
NO#

Characteristics include

(i) Has Simple Common Substrate

(ii) Build OS on top of hardware which enables easy deployment of networking applications.

Router Example



Basic job of router is recieving packets, Checking routing table, forwarding packets out. For building routing table, 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 Seperate data and control plane based architecture of SDN. Since Control plane is Centralized and its on top of all software, hardware it is a network Substrate.

Navyasree
18O1C806
CS358
1 of

(11) A: 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 & flexble. It provides better network Contrd for cloud Computing Service providers to respond quickly to ever changing bussiness requirement & prefered over traditional architecture.

SDN has 3 layers

application layer, Control layer, Infrastructure layer.

Data plane Consists of network elements, which explose their Capabilities to the Control plane via South bound interface

Control plane Sits in middle & has following functions

Navyasree
1801CS06
CS358
Lab1

SDN Architecture diagram

APPLICATION LAYER

Network Control applications

routing ... access Control load balance

SDN northbound interface

CONTROL LAYER

A-CPIs Application-controller plane interface

Interface, abstractions for network Control apps.

Network Graph | Restful API | ..... | intent

Stastics .... Flow tables

Network-wide distributed, robust State management

Link-State info | Host info .... Switching info

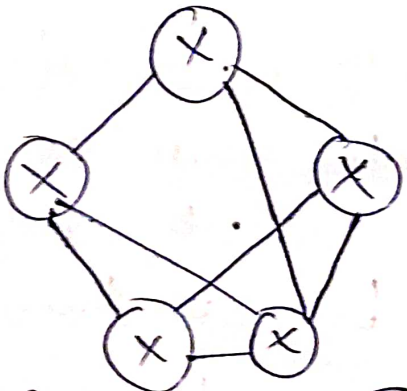
Openflow .... SNMP

Communication to/from Controlled devices

D-CPI : Data Controller plane interface

SDN Southbound interface

DATA LAYER



SDN Controlled Switches

- translate applications requirements and exerts low-level Control over network elements
- provide network information to applicants
- Orchestrate different applications.

Data plane

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. Core functionality of data plane is

- programmatic Control of all functions offered by network element.
- Capability advertisement
- Event notification.

Control plane

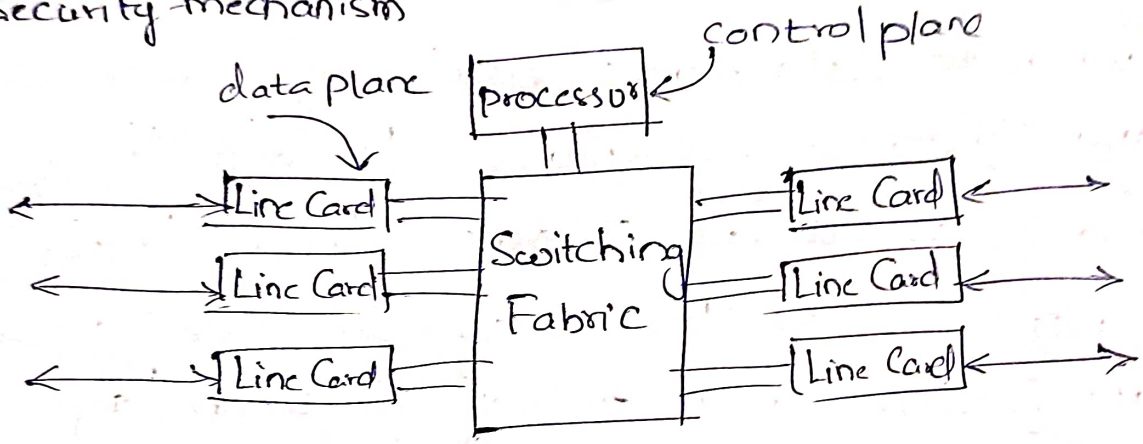
Control Layer is Considered as the brain of SDN. It Communicates with the data plane using Southbound Api's. The intelligence to this layer is provided by Centralized SDN Controller Software. This Controller resides on a Server &

manages policies and flow of traffic throughout network.

Core functionality of Control plane is

- Topology and network State information
- Device discovery
- Path Computation.

Security mechanism



Interaction b/w Data & Control planes

Application plane

SDN Application layer Contains network applications or functions Such as intrusion detection System, load balancing or firewalls. It Communicates with Control layer using northbound API. .....

Applications Specify the resources and behaviors required from network, with Context of business & policy agreement. Applications are developed by Specific programing languages & may need to orchestrate multiple-Controllers to achieve the objectives.

(12) A:

Data Center transport impairments & requirements

The shallow packet buffers cause 3 specific performance impairments

1. 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 creats incast at queue of switch port connected to aggregator

We capture incast instances via packet-level monitoring. Since size of each individual response is small, each individual response leads to loss of packet almost invariably results in a TCP timeout. Thus, whenever time out occrs response almost always misses aggregators deadline.

2. 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 & long flows traverse same queue. Even when no packets are lost, short flows experience increased latency as they are in queue behind packets from large flows, this queue build up impairment occurs & this traffic pattern occurs more frequently

3. Buffer pressure

It is very common for short flows on one port to be impacted by activity on any of many other 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 resource, the queue

build up reduces the amount of buffer space available to absorb bursts of traffic from partition/Aggregate traffic. We terms this impairment buffer pressure. The result is packet lost and timeouts as in incast but without requiring synchronized flows.

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

> Measurements from 6000 Server production cluster
> Instrumentation passively collects logs
- Application-level
- Socket-level
- Selected-packet-level

> More than 150 TB of compressed data over a month

Where as in the case of facebook, it uses Partition / Aggregate

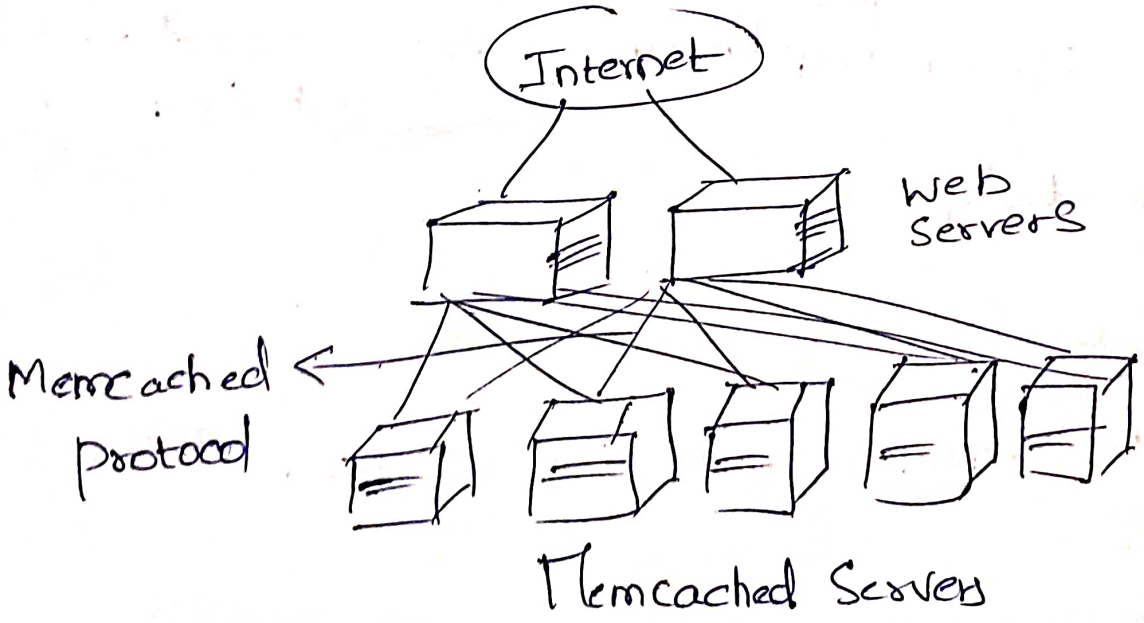
Partition / Aggregate – Foundation for many large scale web apps

Face book

> Uses partition / Aggregate ~ Multiget

Aggregators : Webservers

Workers : Memcached Servers



Workloads are (facebook)

> partition / Aggregate → Delay Sensitive
(query)

> Short messages → Delay Sensitive
[50 KB, 1MB]
(Coordination, Control State)

> Large flows [1MB - 50MB] → Throughput Sensitive.
(Data update)

(13) A:

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 is to

(i) Recieve a packet
(ii) Observe packet fields
(iii) Apply algorithms
(iv) Optionally edit the packet
(v) Forward or discord the packet

Features:

⇒ Flexible
- forwarding behaviour Specified

by Control plane
- Built from basic set of forwarding primitives

⇒ Minimal
- Streamlined for Speed and low power
- open and not Vender Specific

Openflow is an example of forwarding abstraction

Openflow forwarding abstraction

- protocol independent
  - (i) Construct ethernet, IPV4, VLAN, MPLS
  - (ii) Construct new forwarding methods
- Backward Compatible
  - (i) Run in existing networks
- Techology independent
  - (i) Switches, routers, wifi APs
  - (ii) Cellular basestations
  - (iii) WDM / TDM Circuits

b) Network functions virtualization

It is a way to virtualize network services, such as routers, firewalls, and load balances. 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 standard servers instead of proprietary ones. It is one of the primary components of a telco cloud, which is reshaping the telecommunication industry.

With NFV, we don't need to have dedicated hardware for each network function. NVF improves scabbility and agility by allowing service providers to deliver new network services and application on demands without requiring additional hardware resources.

NVF architecture:-

* Virtoalized network functions: Those are software applications that deliver network

Navya Sree
1801C506
CS358

functions such as file sharing, directly services, and IP configuration.

* Network function Virtualization infrastructure: It consist of the infrastudure components - compute, storage, networking.

* Managment, automation and network orchestration: It provides a framework for managing NFV and VNF's.

Benifits of using NFV:

with NFV, service providers can run network functions on standard hardware, instead of dedicated hardware.

NVF gives providers the flexibility to run VNFs across different servers or move them around as needed when demand changes. This flexibility lets service providers deliver services and app faster. For example. If a customer request a new network function, they can spin up a new VM to handle that request. If the function is no longer needed, the VM can be decommissioned. This can also be a low risk way to test the value of a potential new service.

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 fire walls, network address translation (NAT) and intrusion protection. Network operators can use this to set up 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 security, 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 set up to handle traffic flows for connected services. For example, an SDN controller could take a chain of services and apply them to different traffic flows depending on the source, destination, or type of traffic.

The chaining capability automates what

traditional network administrators do when they connect up a services of physical L4-7 devices to process incoming and outcoming network traffic. which traditionally may require a no of manual steps. service chaining is being included in many SDN and network function virtualisation use cases and deployments. including data cirkess. Another benifit is when used in combination with SDN. IS optimizing 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 chainning is facing a new networking technology that is purposefully trying to get away from service chaining. That technology is called SASE. The chain in service chainning represents the service that can be connected across the network using software provisioning. This is espicially important in NFV world.

Navya Sree
1801cs06
CS358

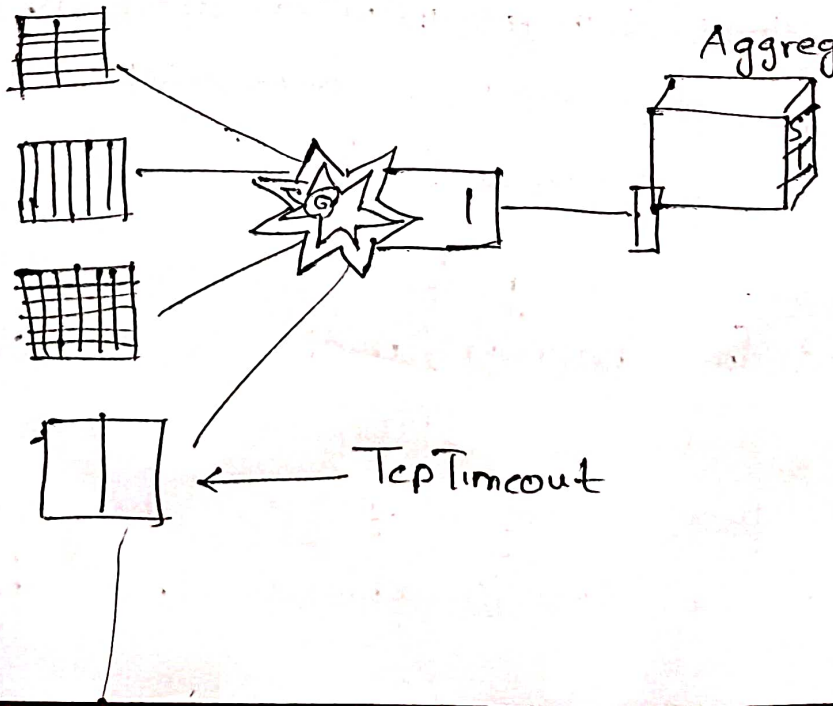
The Technology's capabilities mean that a large number of virtual network functions can be connected together in a NFV environment. Because its done in softwares using virtual circuits, these connections can be set up and form down as needed with service chain provisioning through the NFV.

Navyasree
1801cs06
cs358

14) A:

Incast problem in data Centres

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 transmits data blocks to a Single recieves and any Sender is not allowed to Send the next block untill all students finish transmitting Therefore in the delay-Sensitive Online application which employ partition/aggretation Computng. Very few Senders extended the time, So the System responses are included from final result due to missing aggregators deadline.



Navyasree
18D1CS06
CS858
ref

Q4: Solutions recommended to solve TCP incast problem

1) Adjusting System parameters

It improves TCP incast problem by adjusting parameters, (decreasing) 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

→ Reduce RTOmm to reduce throughput Collapse

→ Shrinking MTU (max. transmission limit)

3) Replacing loss - bossed TCP Version

→ Adjusting Congestion windows properly according to delay information generated by RTT measurement. Slight adjustment of window size is beneficial to protect the buffer

4) New Transmission protocols.

DCTCP Algorithm: It works because

(i) High Burst tolerance

→ Large buffer headroom → burst fit

→ Aggresive making → Sources react before packets are dropped.

Navyasree
1801CS06
CS358

(ii) low latency

Small buffer Occupancies → low quality delay

(iii) High throughput

ECN agency → Smooth rate adjustments, low variance

Navyasree
1801es06
cs358
ref

(15) A: Markov Decision problem

These processes are discret-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.

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 action Space

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

(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 recieved after transmitting from State s to s' due to action a.

A policy function Π is a probabilistic mapping from State Space to action Space.

Navyasree
18O1CS06
CS358

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 maximize the expected return.

$$\pi_*(s) = \arg\max_a \left\{ \sum_{s'} p(s'|s,a) \left(R(s'|s,a) + \gamma V(s') \right) \right\}$$

We can Coverage 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.

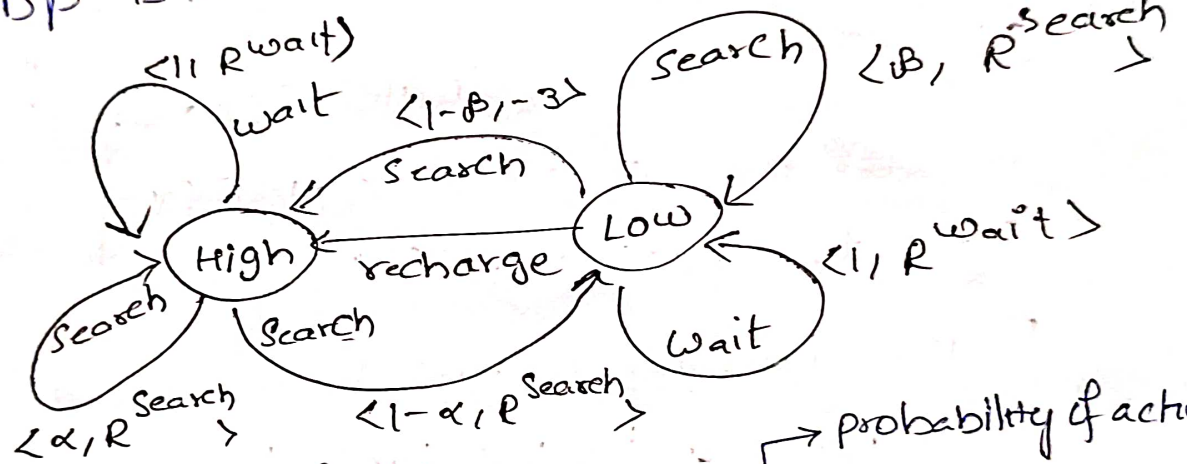
Both recursively update a new estimation of Optimal policy & State Value using an older estimation of those values.

Ex:- Consider 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

MDP is as follows,



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 Value of action in particular State. It does not require a model of environment and it can handle problems with Stochastic transitions & rewards without requiring adaptations.

Navya Sree
1801CS06
CS358

For any finite MDP (FMDP), Q-learning finds an Optimal policy in the Sense of maximizing expected value of total reward Over all Succesive steps, Starting from Current state. Q-learning Can identify an Optimal action-Selection policy for any given FMDP, given infinite exploration time and a partly random policy.

Objective: We estimate Optimal Q-function we assume a Q-table, mapping State action Pairs to Values ie..) $Q : S \times A \to R$

Before learning begins, Q is initialized to a possibly arbitary values. Then at each tme 't' the agent selects an action a_t, Observes a reward 'r_t', enters new state S_{t+1}, Q is updated.

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

- $Q'(s_t, a_t)$: new value
- $Q(s_t, a_t)$: Old value
- α : Learning rate
- r_t : reward
- γ : discount factor
- $\max Q(s_{t+1}, a)$: future expected
- $Q(s_t, a_t)$: old value

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

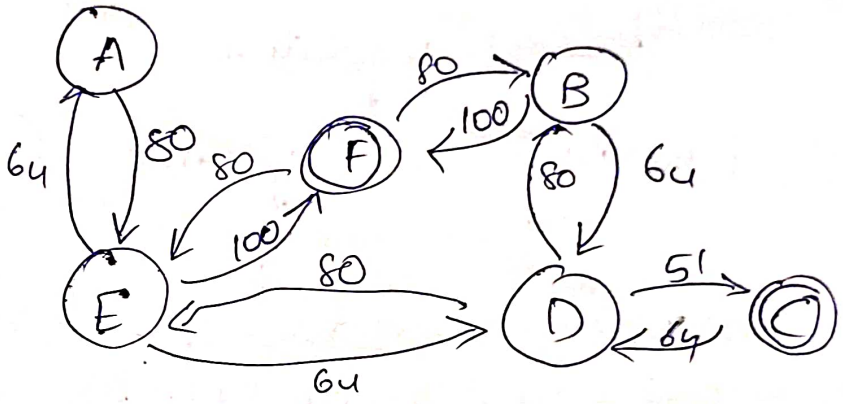
Three factors,

i) $(1-\alpha)\,Q(s_t, a_t)$ → Current Val weighed by α

ii) αr_t : Reward if a_t is taken when

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

Ex: Consider following environment



Initial Q table,

$Q =$

	A	B	C	D	E	F
A	0	0	0	0	0	0
B	0	0	0	0	0	0
C	0	0	0	0	0	0
D	0	0	0	0	0	0
E	0	0	0	0	0	0
F	0	0	0	0	0	0

Final Q table,

$Q =$

	A	B	C	D	E	F
A	–	–	–	–	80	–
B	–	–	–	64	–	100
C	–	–	–	64	–	–
D	–	80	51	–	80	–
E	64	–	–	64	–	100
F	–	80	–	–	80	100

Navya Sree
1801CS06
CS358

MDP	Q-Learning
(i) Non-temporal difference	(i) Temporal difference
(ii) Discrete Time Stochastic Control process	(ii) Continous Time Stochastic control process
(iii) Model based / policy based algorithm	(iii) Model free Algorithm
(iv) utilizes Dynamic Programming	(iv) utilizes Bellman Equation
(v) uses value function & policy function for optimization	(v) Utilizes Q-table for Optimization
(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 ie. when 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) A:

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 2 things that are useful for agent to do

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

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

Trade off: During learning, it might be the case that the agent chooses the action value which has the highest estimate (greedy action) or Something else (non-greedy action),

Doing only one of these doesn't help in Convergence key is balancing b/w exploitation & explanation. Only exploitation might be good in short run but might miss long sum optimum. ∴ even though initially exploration gives bad results it will help in convergence

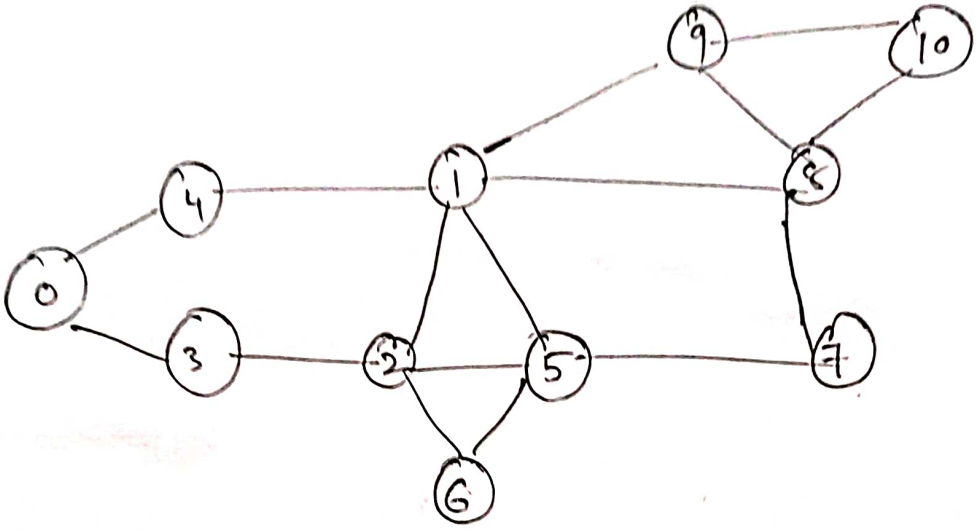
Few algorithms that balance b/w both are

1. Greedy Algorithm
2. upper Confidence bound
3. Thompson Sampling

Note : The property that needs to be held by exploitation is "Degradation". Exploration must happen in the initial stages of learning & must die down ~~extensa~~ eventually. or else it accounts to high Variance

Eg: Decaying ε Greedy policy

(19) A: Given Graph,



It is also given, Optimal Q-matriv

As we know, $\Pi^*(s) = \underset{a}{\text{argmax}}\,(Q(s,a))$

$\Rightarrow$ $\Pi^*(0) = 4$ (138 is max val in 0 row)

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

$\Pi^*(1) = 8$ (or 9) (218 " " " 1 row)

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

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

also possible path is $0 \to 4 \to 1 \to 9 \to 10$