Test - II
19/1/18

El-309 IInd mid term test (computer communication)

Q.1 mac layer के pure Aloha protocol को समझाइये।

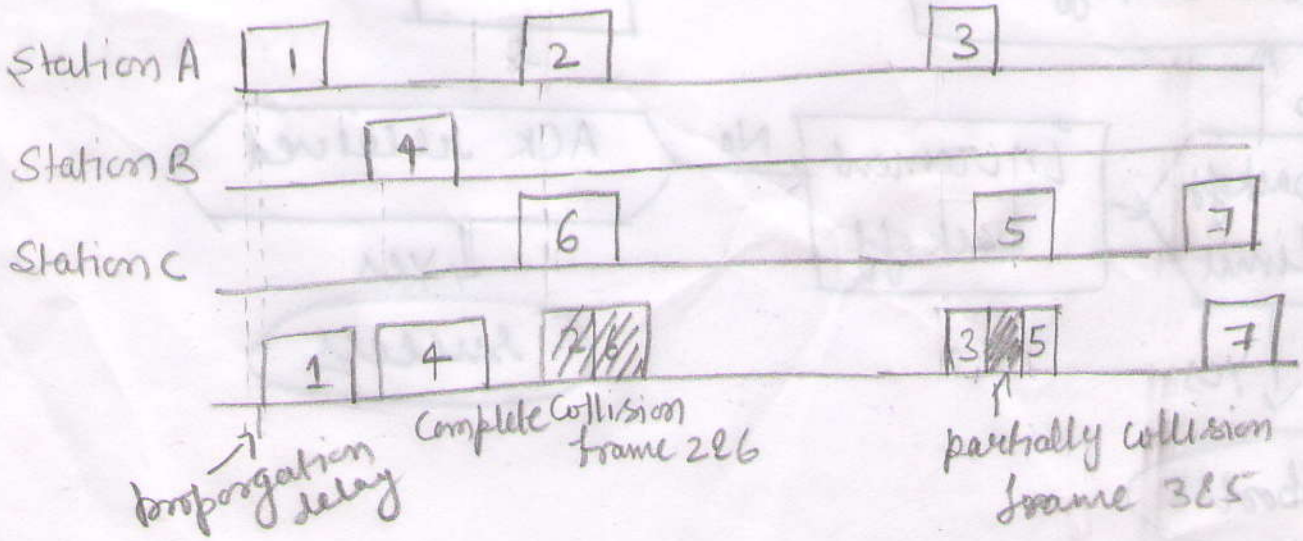
Ans. Sliding window protocol को समझाइये।

Q.3 routing क्या है? Shortest-path routing को समझाइये।

Q.4 leaky bucket technique को समझाइये।

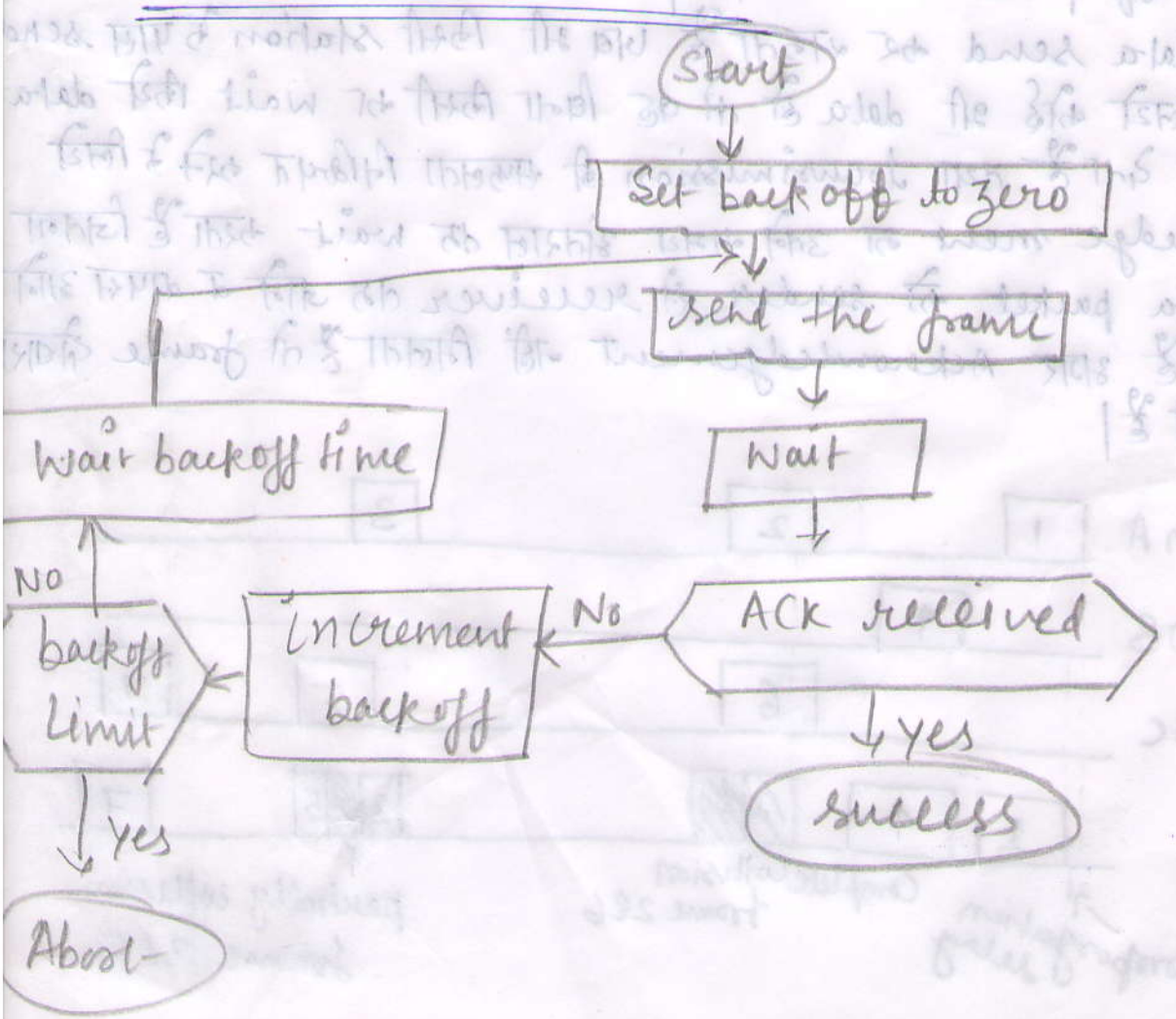
Ans.1 Pure Aloha Protocol :— Aloha को packet radio network के लिये develop किया गया था अब इसे shared transmission media के लिये use किया जाता है वास्तविक ALOHA तकनीक को ही Pure ALOHA कहते हैं।

Concept of pure ALOHA → pure ALOHA में कोई भी station कभी भी data send कर सकता है जब भी किसी station के पास send करने के लिये कोई भी data हो तो वह बिना किसी का wait किये data send कर देता है तथा transmission की सफलता निश्चित करने के लिये Acknowledgement का उतने समय अंतराल तक wait करता है जितना समय data packet को sender से receiver तक जाने व वापस आने में लगता है अगर Acknowledgement नहीं मिलता है तो frame दोबारा भेजी जाती है।



दिये गये चित्र मे pure ALOHA scheme दिखाई गई है जिसमें यह माना है कि सभी frame same length के है तथा प्रत्येक को 1 time unit transmission के लिये चाहिये। यहाँ तीन station A, B, C frame transmit कर सकते है station A और C frame 2 and 6 एक ही समय पर transmit करते है तथा Collission के कारण दोनो frame पूरी तरह corrupt हो जाती है इसी प्रकार station B और C की frame 3 और 5 partially collide करती है लेकिन corrupt हो जाने के कारण इन दोनो frames को भी दोबारा transmit करना पड़ता है। अगर किसी frame की 1 bit भी दूसरी frame से collide हो जाये तो दोनो frame corrupt हो जाती है व दोनो को दोबारा भेजा जाता है।

Flow chart of pure ALOHA :-



Ans 2 Sliding window Protocol :- Sliding window flow control का एक method है जिसमें sender बिना Acknowledgement का wait किये कई सारी frames भेज सकता है sliding window में sender व receiver दोनो के पास एक-एक imaginary window होती है जिसका size fix होता है जैसे कि अगर इसका size N है व N = 8 है तो 0,1,2,3,4,5,6,7 नंबर की frames sender receiver को भेज सकता है व दोनो सिरो पर window N-1 frame को रख सकती है इसका मतलब sender बिना Acknowledgement का wait किया N-1 यानि (7) frame receiver को भेज सकता है receiver भी प्रत्येक frame का Acknowledge नही भेजता बल्कि अगर वह एक साथ कई frames का Acknowledgement भेज सकता है जैसे अगर frame 0,1,2,3,4 receiver को मिल-चुकी है तो वो Acknowledgement में 5 भेजता है जो कि next expecting frame का number है।

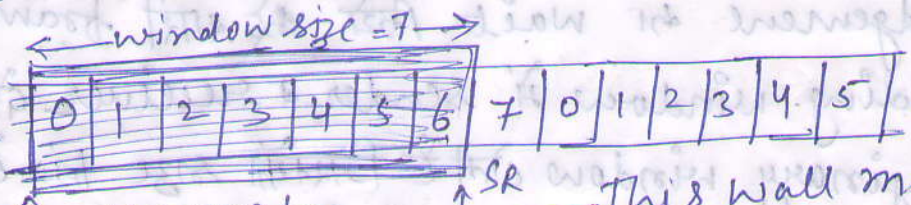
अगर sliding window में receiver को बीच में कोई packet नही मिलता तो sender 2 strategies use कर सकता है।

(1) go back N sliding window protocol
(2) selective repeat window protocol

1) go back N sliding window protocol —

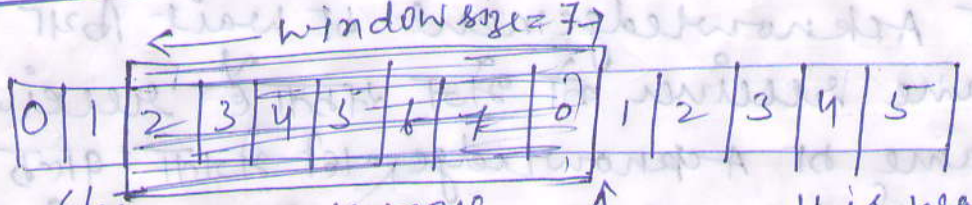
Sender sliding window — Sender side पर जब तक frames का Acknowledgement नहीं आता तब तक उन्हें buffer में store रखा जाता है। window के left में वो frame होते है जिनका Acknowledgement आ चुका है और discard किये जा सकते है जबकि right में वो frame होते है जिन्हें तब तक नही भेजा जा सकता जब तक window उस पर नो

तरफ जाये क्योंकि window का size fixed है। window नये भेजे जाने वाले frames की तरफ सरकती है जब window वाले frames का correct acknowledgement मिल जाता है।



Before sliding.

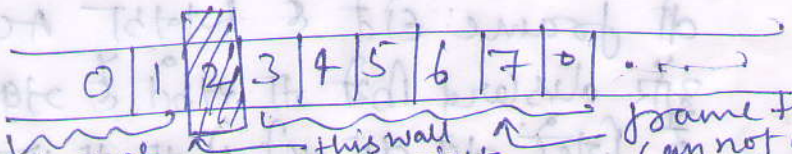
After sliding two frames.



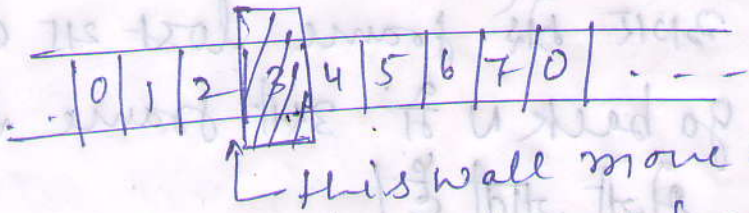
ऊपर वाले चित्र में frame 0-6 भेजे जा चुके हैं लेकिन जैसे ही 0 और 1 का Acknowledgement मिलता है window 2 frame right में slide हो जाती है व अब frame 7 व 0 भी send कर दिये जाते हैं।

receiver side sliding window :- Go back N में receiver side की window का size हमेशा 1 होगा। receiver हमेशा next frame की प्रतीक्षा करता है अगर उसे उसी order में frame नहीं मिलता तो वह आने वाली out of order frame को discard कर देता है इस तरह receiver के पास N frame को store करने के लिये कोई buffer नहीं होता।

a) Before sliding

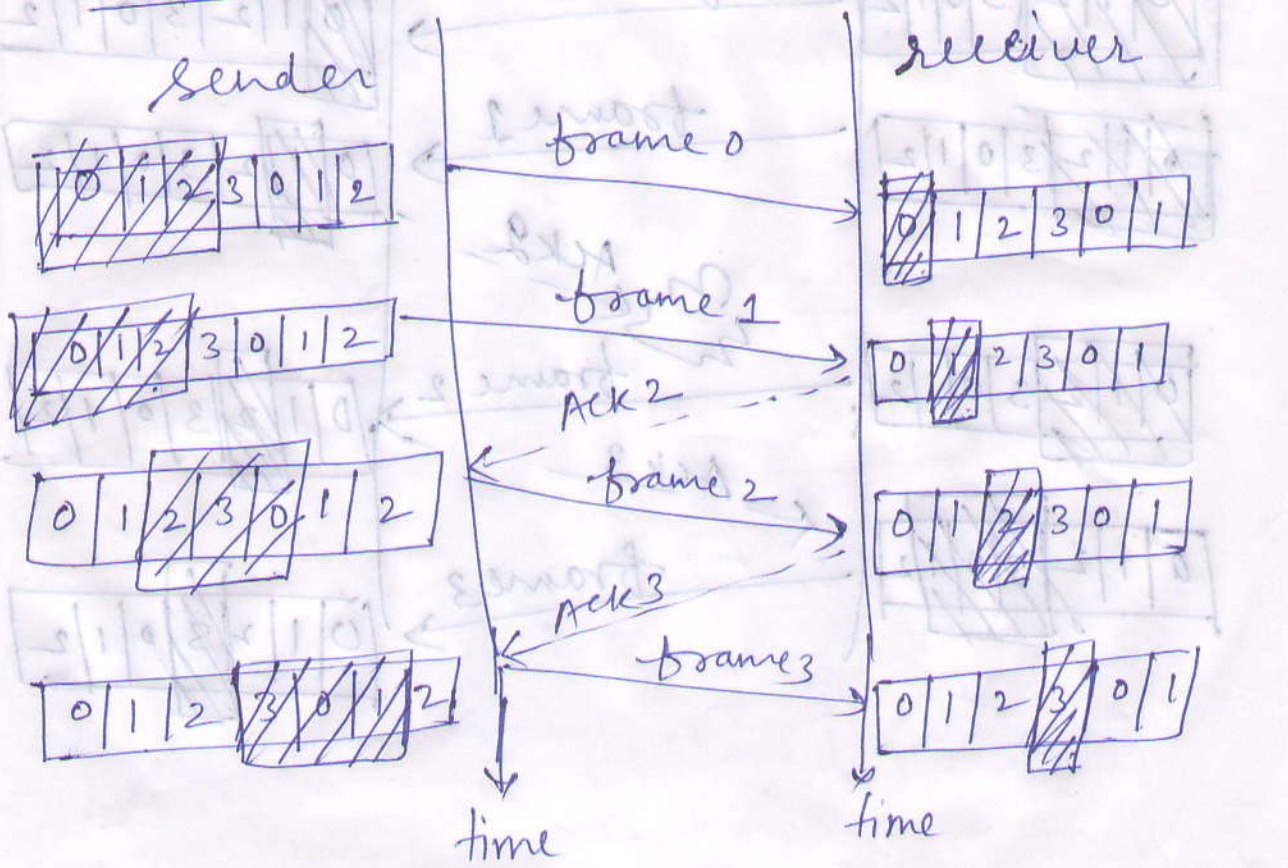


After sliding one frame



Operation of Go back N sliding window protocol

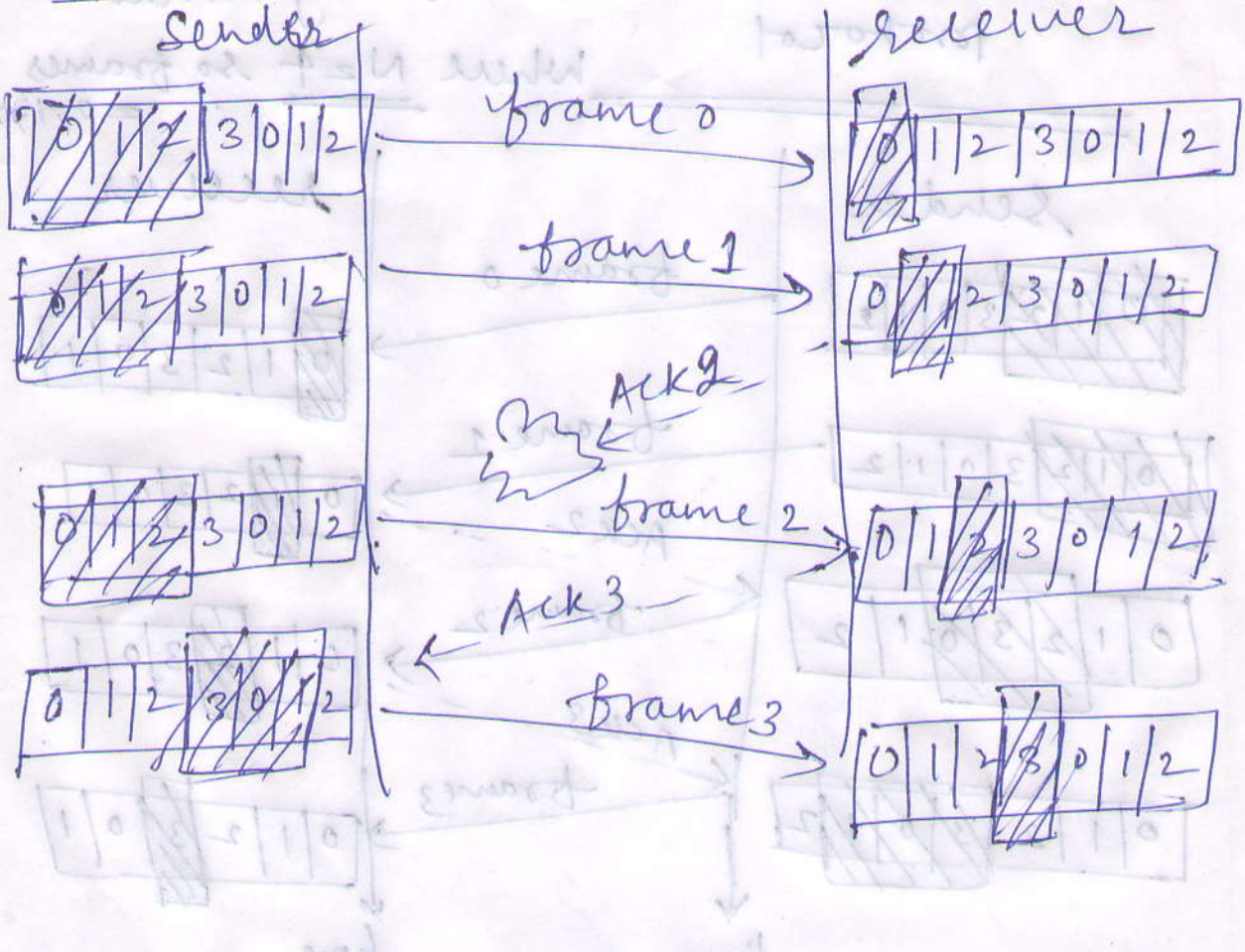
Where N = 4 so frames = 0, 1, 2, 3



Damaged or lost frame →

अगर कोई frame lost या damage हो जाती है तो go back N में उस frame से दोबारा सारी frames भेजी जाती है।

Delayed or lost Acknowledgement →



Ans 3. Routing: Network layer का प्रमुख कार्य किसी packet के लिये source machine से destination machine तक का route तय करना होता है क्योंकि Network में अलग-2 packet अलग-2 route से होते हुये अपने destination तक पहुचता है routing Algorithm यह deside करती है कि आने वाला packet किस output line पर transmit होगा। routing Algorithm दो प्रकार की होती है—

1. NonAdaptive Algorithms — इस तरह की Algorithm में प्रत्येक source-destination जोड़े की cost पहले से निश्चित होती है तथा route booting के समय ही fix कर लिया जाता है।

2. Adaptive Algorithm — इसमें route fix नहीं होता, तथा network topology व traffic के अनुसार routing decision change हो सकता है।

Shortest Path Routing → Shortest Path routing एक Non Adaptive type की Algorithm है। इस Algorithm में network को एक graph की रूप में represent करते है जहाँ प्रत्येक vertex, router को तथा प्रत्येक edge एक communication link को represent करते है किन्ही दो router के बीच का route decide करने के लिये Algorithm उनके बीच का shortest path तलाश करती है। Shortest path (distance, bandwidth, average traffic आदि से cost compute करते है) के लिये प्रत्येक edge को weight assign करना होता है जो कि cost पर depend करे Shortest path routing के लिये Bellman-ford Algorithm व Dijkstra Algorithm use की जाती है।

Dijkstra Algorithm: —

V = set of nodes in the network

S = source node

$l(w,v)$ = link cost from node w to node v, the cost is ∞ if the nodes are not directly connected.

$D(n)$ = least cost, from s to n nodes that is currently known to the algorithm.

- The steps for the Algorithm –

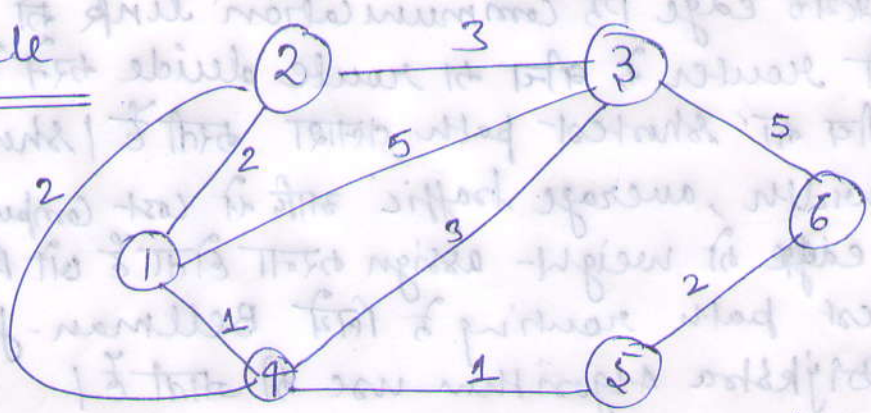
1. Initialize
   set $D[s] = 0$
2. for each node in V except source node
   – do $D[v] = l[s, v]$
3. while $S \neq V$ do // select –
4. begin
5. choose a node w in $V-S$ such that $D[w]$ is a minimum
6. Add w to S;
7. for each in $V-S$ do // iterate for all nodes to recompute the cost

$$D[v] = \min(D[v], D[w] + l[w, v])$$

end

end.

Example



A simple network with associated link cost

initialize $S = \{1\}$, $D[1] = 0$ and $D[i]$ is $2, 5, 1, \infty, \infty$ for node 2, 3, 4, 5 and 6 respectively.

अगले iterationमें minimum link cost ~~वाली~~ node select करेंगे जो कि node 4 है और वापस से cost-

Algorithm की line 3-7 से Compute करेंगे / हम सभी iteration को table की form में दर्शा रहे हैं व last iteration में हमें source से प्रत्येक node तक का shortest path मिल जायेगा |

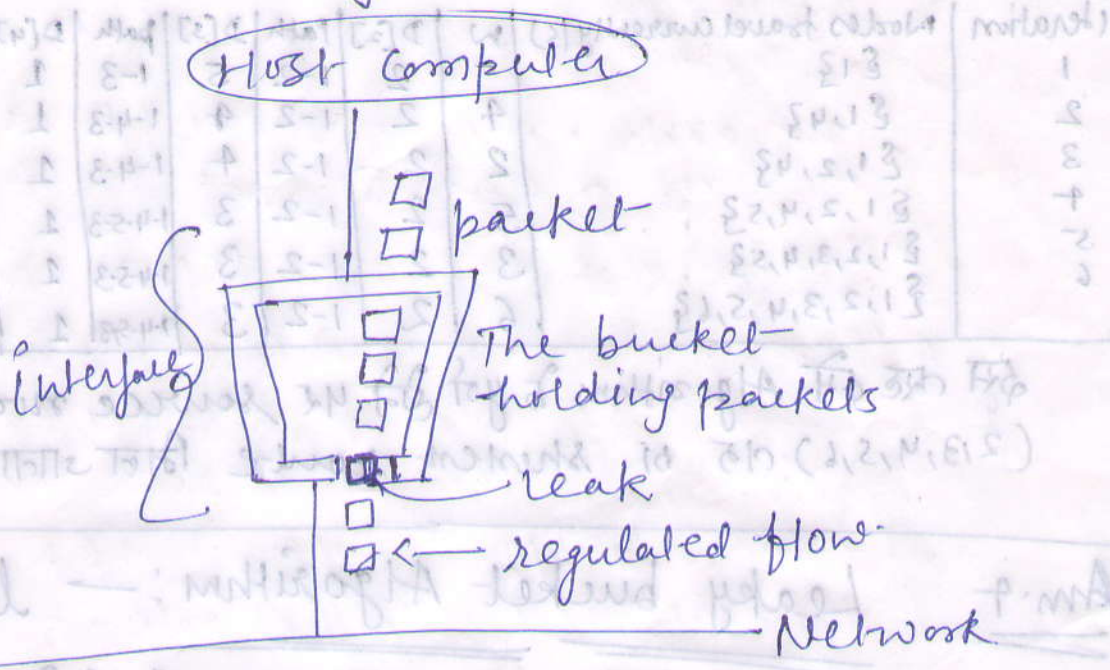
iteration	Nodes travel currently (S)	W	D[2]	Path	D[3]	path	D[4]	path	D[5]	path	D[6]	path
1	{1}	-	2	1-2	5	1-3	1	1-4	∞	—	∞	—
2	{1,4}	4	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	—
3	{1,2,4}	2	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	—
4	{1,2,4,5}	5	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-
5	{1,2,3,4,5}	3	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-
6	{1,2,3,4,5,6}	6	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-

इस तरह हमें Algorithm के पूर्ण होने पर source node 1 से प्रत्येक node (2,3,4,5,6) तक का shortest route मिल जाता है |

Ans. 4 Leaky bucket Algorithm :— leaky bucket Algorithm Congestion Control के लिये use की जाती है यह traffic shaping approach पर based है, traffic shaping एक approach है जिसमें Congestion को manage करने के लिये data transmission को एक average rate पर maintain किया जाता है |

यह Algorithm एक ऐसी bucket में पानी डालने पर आधारित है जिसके तले में छेद है इस bucket की एक निश्चित गहराई है तथा यह एक Constant rate पर leak करती है जब यह खाली नही होती | इस bucket की गहराई की वजह से पानी के बहाव की असमानतायें अवशोषित हो जाती हैं और पानी निकलने की दर Constant ही रहती है | अगर पानी का बहाव कभी ज्यादा (अचानक बहुत) व कभी कम है तो हमें अधिक गहरी bucket व अगर पानी का बहाव normal है तो कम गहरी bucket की आवश्यकता होती है पानी को एक Constant speed

bursty है तो bucket deep व अगर network smooth है तो shallow bucket की आवश्यकता होती है तथा packet निकलने की दर उस traffic rate के बराबर होती है जितना हम network की average data rate चाहते हैं।

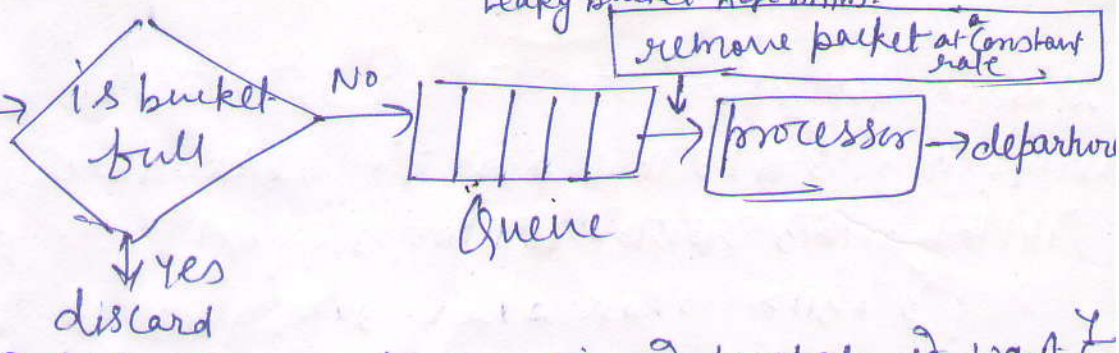


Leaky bucket Algorithm

Leaky bucket Algorithm के दो variations use किये जाते हैं। (1) जब packet size fixed हो।
(2) जब packet अलग अलग size के हो।

(1) <u>जब packet size fixed है</u> :— इस तरह की leaky bucket को implement करने के लिये प्रत्येक host के पास internal queue buffer होता है व एक counter queue के content को record करता है अगर counter सीमा से ज्यादा हो जाता है (यानी bucket full हो जाती है) तो counter change नहीं होता और packet को discard कर दिया जाता है लेकिन अगर bucket full नहीं है तो packet को buffer मे store कर लेते है counter को increment कर देते है इस प्रकार यह offer किये जाने load को सीमित कर congestion को control करती है जब तक bucket खाली नहीं होता ये एक continuous rate पर leak होता रहता है

Leaky bucket Algorithm.



Ans

चित्रानुसार एक FIFO queue, fixed size के packet को रखती है तथा प्रत्येक clock tick पर queue से एक packet निकालता रहता है अगर traffic variable length के packet रखता है तो output rate packet के number पर आधारित ना होकर bytes के number पर आधारित होती है।

(2) <u>packet अलग-अलग size के हो</u> → इस प्रकार की leaky bucket algorithm में counter को maximum पर initialize करते है। अगर आने वाले packet का size counter के size से कम है तो packet को send कर देते है व counter को उस packet के size जितना घटा देते है जब तक counter का size आने वाले packet से ज्यादा होता है तब तक packet send करते रहते है व जैसे ही counter का size कम हो जाता है packet को discard कर देते है व अगली clock tick पर counter को फिर से maximum bytes की संख्या से load करते है व process दोहराते है। इसे flow chart से भी दर्शा सकते है →