SDN Based prioritized bandwidth scheduling using Reinforcement learning

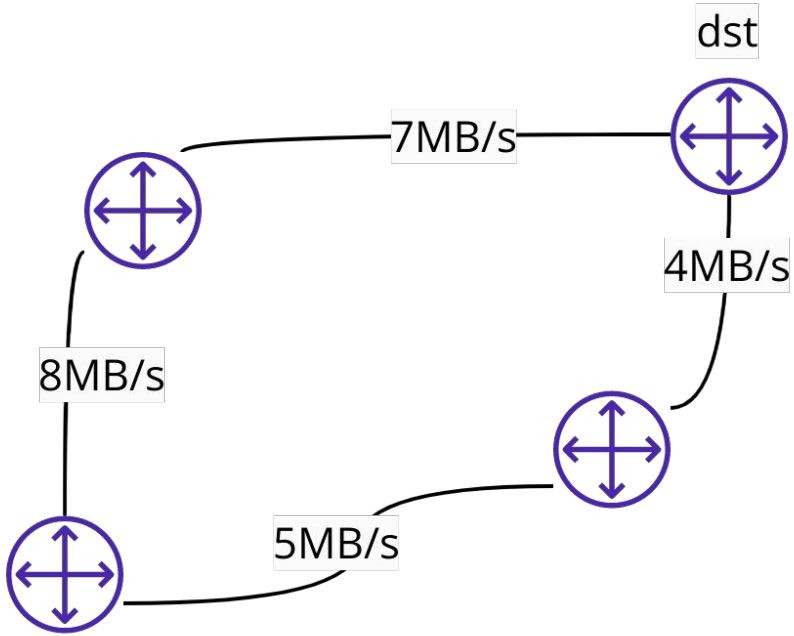
Manoj S (2019506048)

Aravinth Kumar A M (2019506012)

Vimal V(2019506114)

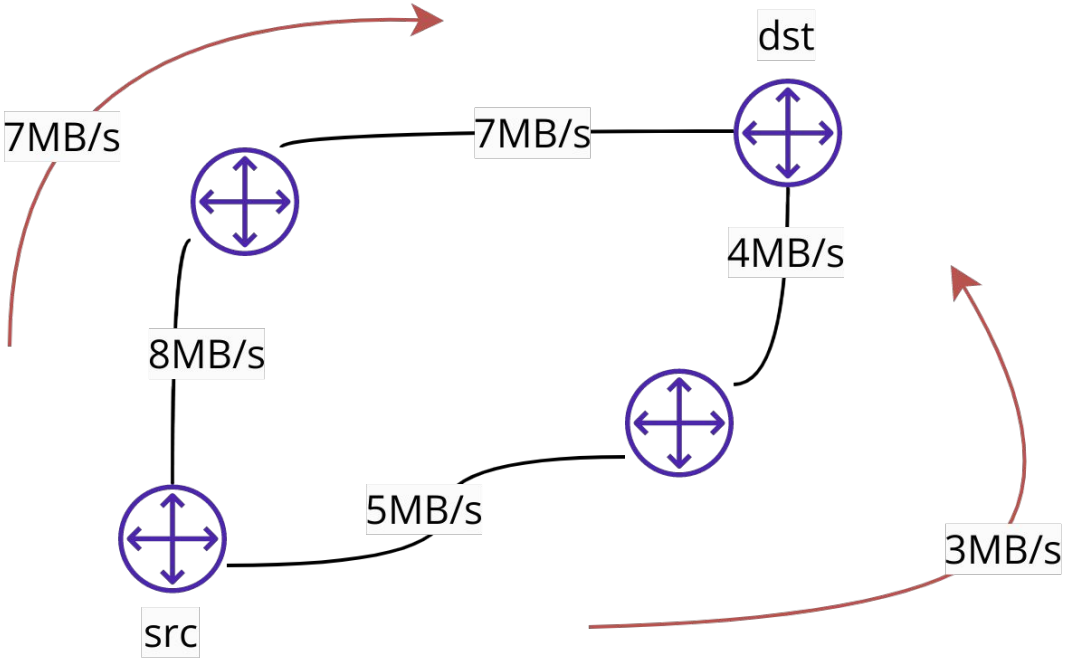
Supervised by

Dr. S Umamaheswari



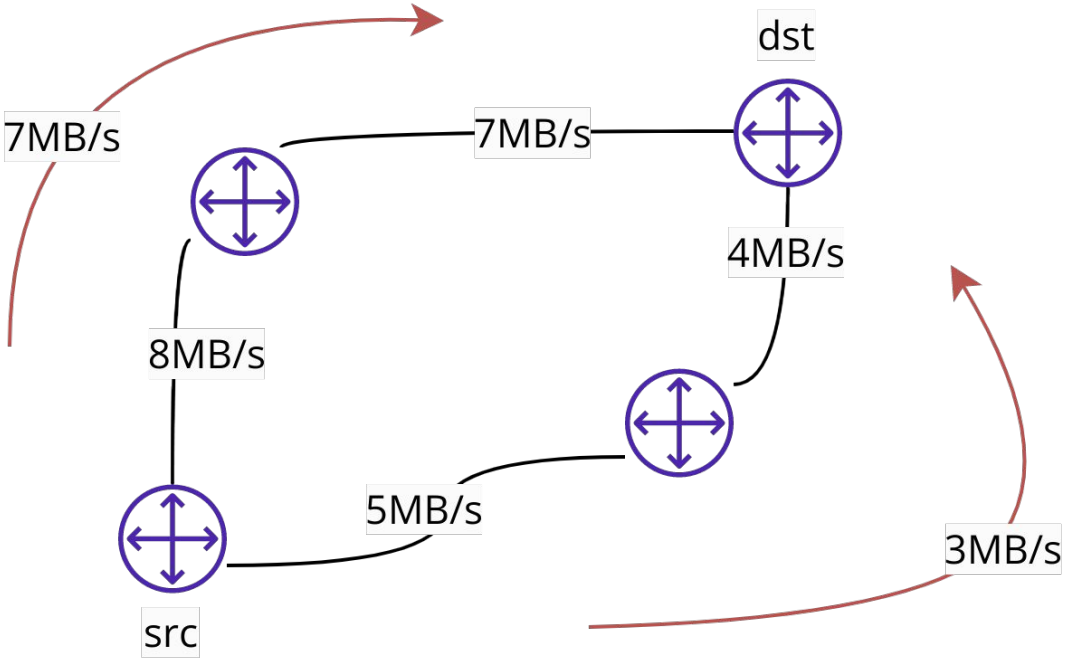
Maximum 7MB/s can be allotted using traditional algorithms

10MB/s required



Using group services we can split the bandwidth

10MB/s required



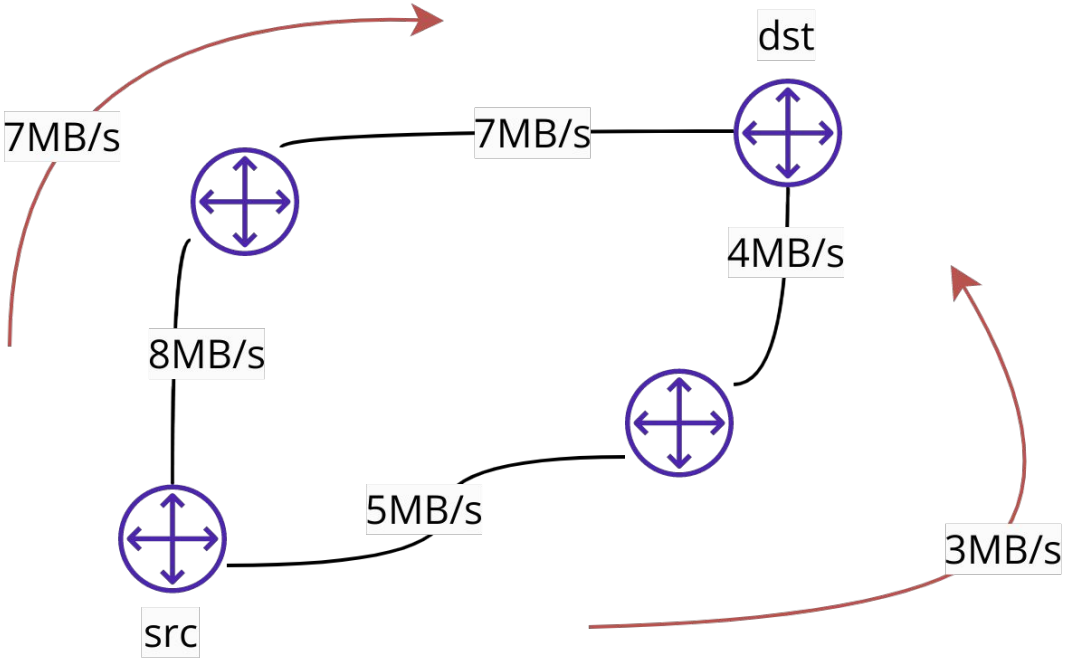
10MB/s required

Using group services we can split the bandwidth

There are 2 problems

Which path to choose

% to split for each path



10MB/s required

Using group services we can split the bandwidth

There are 2 problems

Which path to choose

Reinforcement Learning

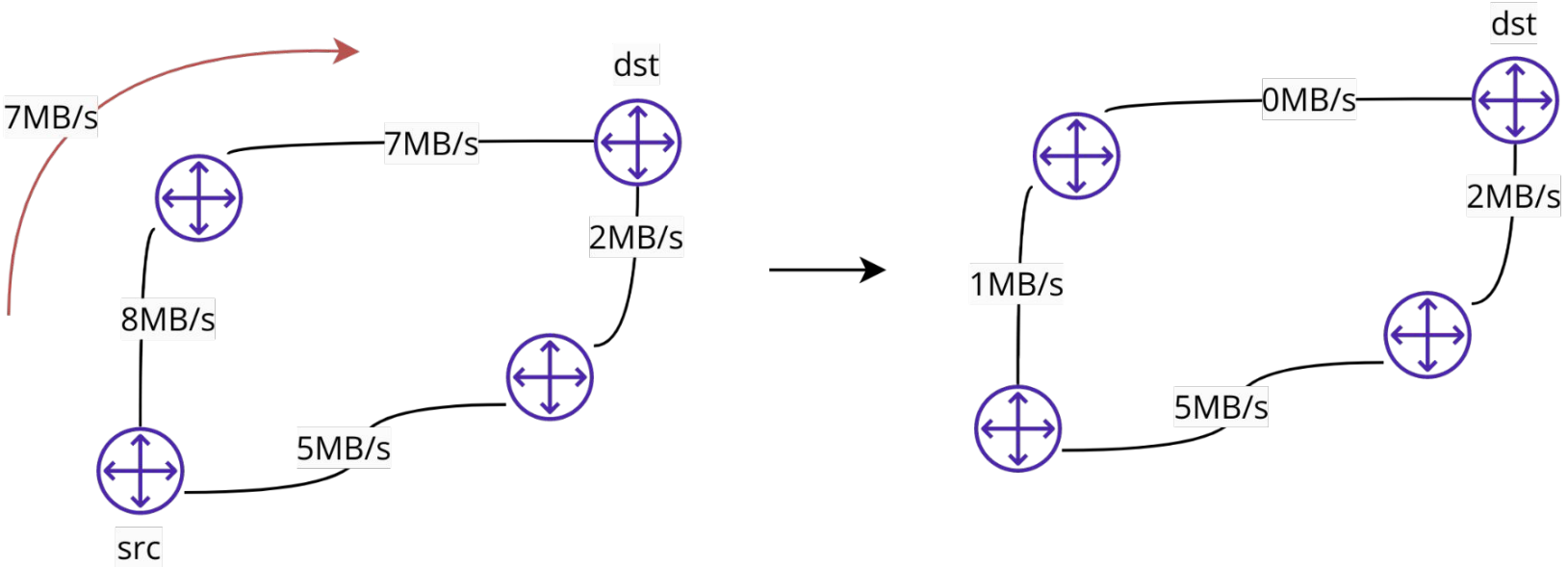
% to split for each path

Linear programming

Designing Inputs / state

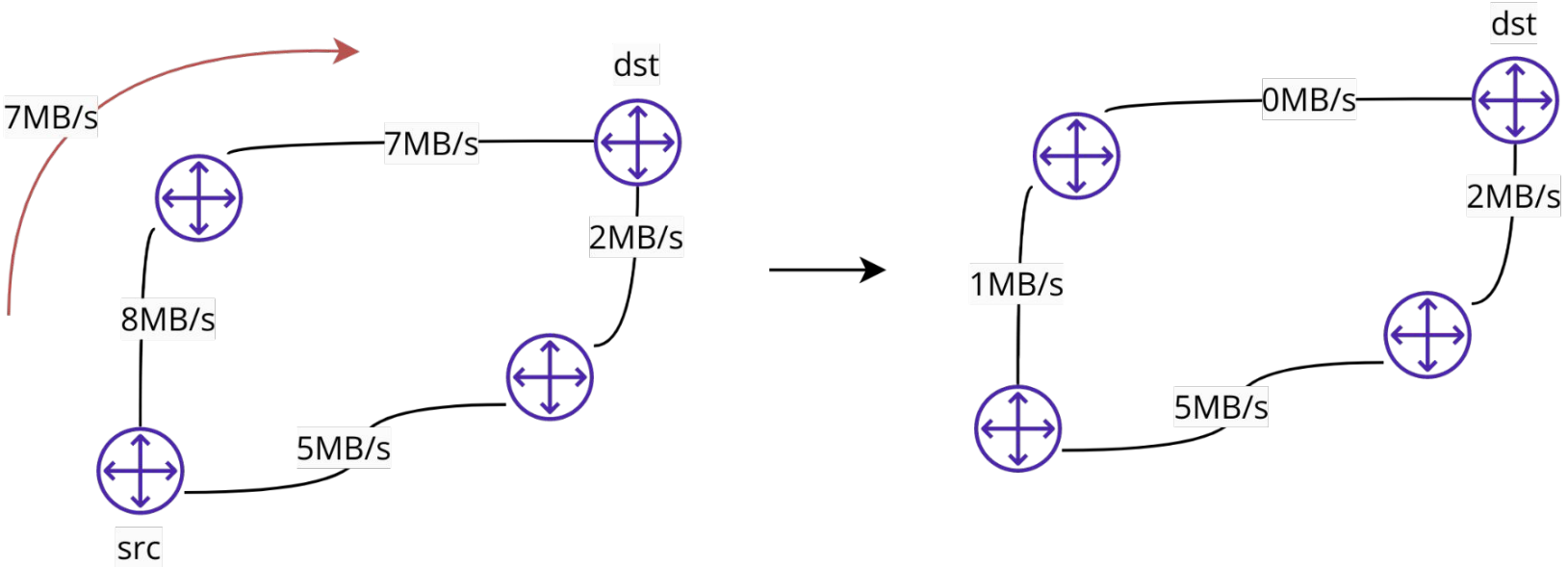
Designing Inputs / state

Fix on Action



Designing Inputs / state

Fix on Action

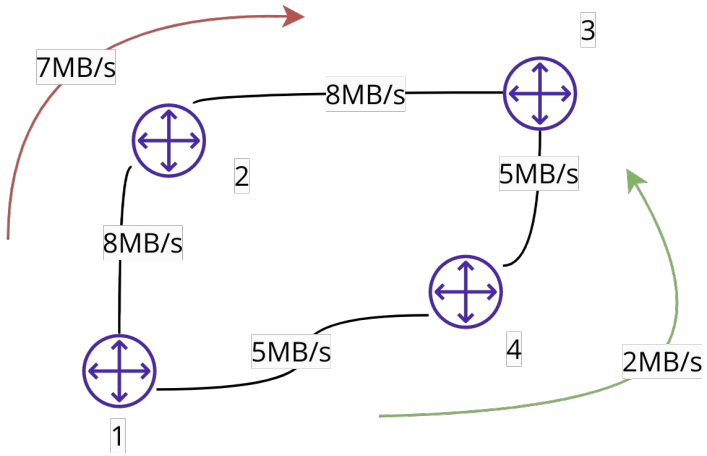


state1

Change in residual bandwidth is captured as state

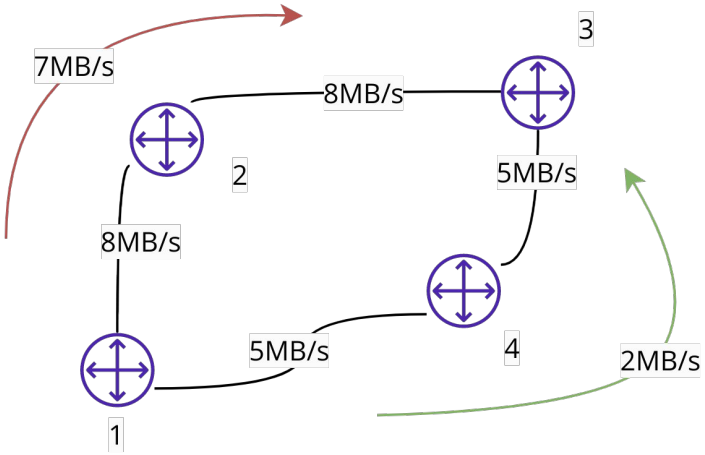
state2

Rewards

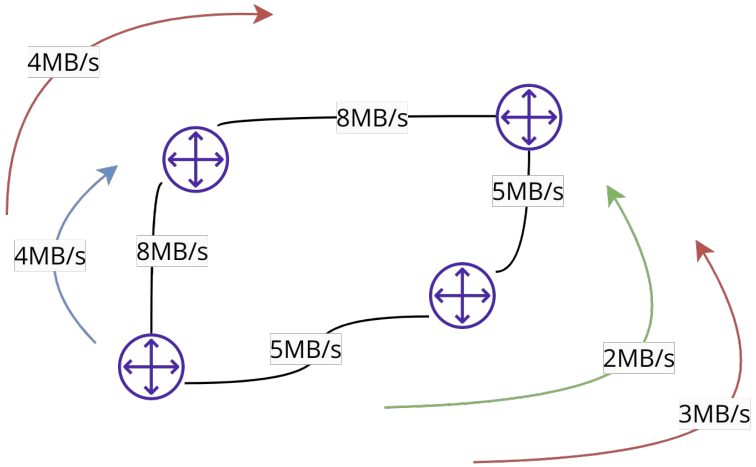


Another 4MB/s
from 1 to 2 ?

Rewards



Another 4MB/s
from 1 to 2 ?



Rewards

	Technique 1	Technique 2
Step 1	7	3 + 4 = 7
Step 2	7 + 2 = 9	3 + 4 + 2 = 9
Step 3	9	9 + 4 = 12
Expected at step 1	7 + y*9 + y*y*9	7 + y*9 + y*y*12
	22.39	24.82

Rewards

	Technique 1	Technique 2
Step 1	7	3 + 4 = 7
Step 2	7 + 2 = 9	3 + 4 + 2 = 9
Step 3	9	9 + 4 = 12
Expected at step 1	7 + y*9 + y*y*9	7 + y*9 + y*y*12
	22.39	24.82

Predict Expected Reward

Choose paths until it crosses certain threshold

Why is Linear Programming required

- Assigning one flow higher bw can cause disruption to many other
- Compromising 1 flow might give higher advantage to many other
- Ability to include **priority**

Why is Linear Programming required

- Assigning one flow higher bw can cause disruption to many other
- Compromising 1 flow might give higher advantage to many other
- Ability to include **priority**

Constraints in Linear Programming

- For every request the sum of bandwidth assigned to multiple paths should be less than requirement
- For every link the sum of bandwidths assigned from different requests and paths should be less than capacity

$Let,$

$l\ be\ the\ number\ of\ requests\ arrived$

$m\ be\ the\ number\ of\ all\ possible\ paths$

$k\ be\ the\ total\ number\ of\ links$

$F_{ijk} = 1\ \ if\ in\ the\ i^{th}\ request\ j^{th}\ path\ is\ chosen\ and$

$represents\ link\ k\ else\ 0$

$B_{ijk}\ represents\ the\ bandwidth\ of\ the\ link\ to\ be\ assigned$

$$\forall_k \left[\sum_{i=0}^{l} \sum_{j=0}^{m} (B_{ijk} * F_{ijk}) < Capacity_k \right]$$

$$\forall_i \left[\sum_{j=0}^{m} \sum_{k=0}^{n} (B_{ijk} * F_{ijk}) < Required\ Bandwidth_i \right]$$

$$let\ P_i\ represent\ the\ priority\ of\ flow\ i$$

$$max\left(\sum_{i=0}^{l} \sum_{j=0}^{m} \sum_{k=0}^{n} (B_{ijk} * F_{ijk} * P_{ijk}) \right)$$

Neural Network

Training should also be done in java

Neural Network

Training should also be done in java

Neural Network

Training should also be done in java

Replay Buffer would be used to store previous data

```
while (true):
      Replay Buffer -> rb
      for task_i in episode:          // each time new episode
            Buffer -> b
            If random < e:
                  take random action
            else:
                  take action as per neural net
            Store (state, reward) in b

      reward = 0
      for (st,rw) in reverse(b):
            reward = rw + γ*reward
            store( st, reward ) in rb
```

Neural Network

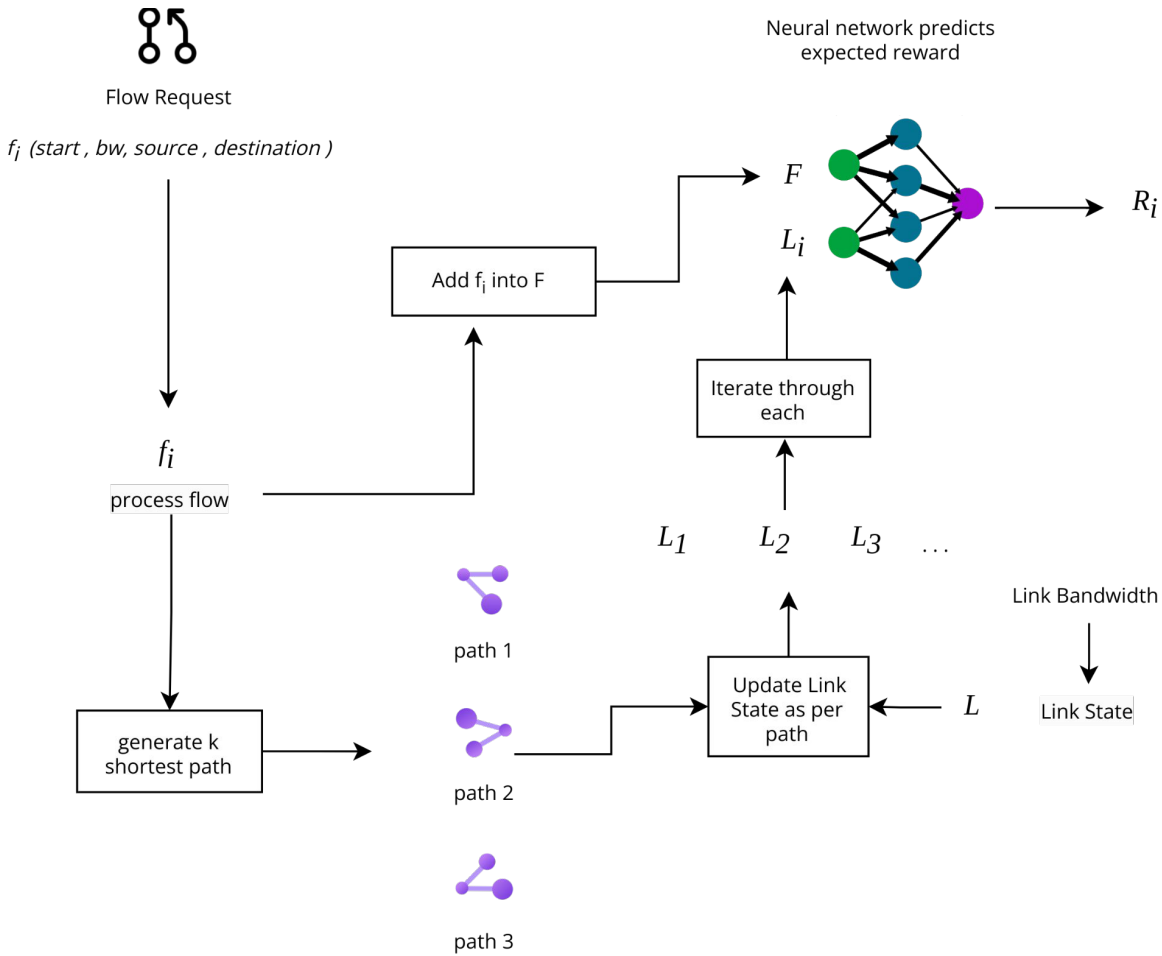
Training should also be done in java

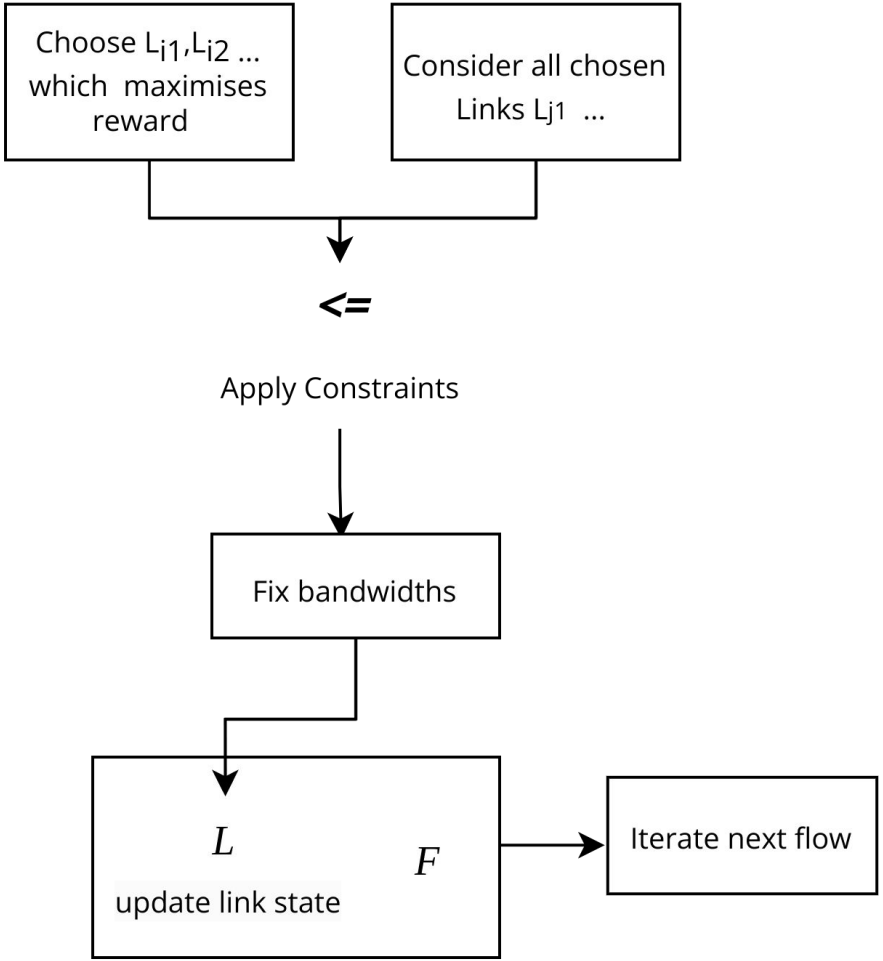
Replay Buffer would be used to store previous data

Training would run periodically by sampling replay buffer

```
while (true):
      Replay Buffer -> rb
      for task_i in episode:          // each time new episode
            Buffer -> b
            If random < e:
                  take random action
            else:
                  take action as per neural net
            Store (state, reward) in b

      reward = 0
      for (st,rw) in reverse(b):
            reward = rw + γ*reward
            store( st, reward ) in rb
```


Flow Request
f_i (start , bw, source , destination)
Neural network predicts expected reward
F
L_i
R_i
Add f_i into F
Iterate through each
f_i
process flow
L_1
L_2
L_3
...
path 1
path 2
path 3
generate k shortest path
Update Link State as per path
L
Link Bandwidth
Link State


Choose L_{i1},L_{i2} ... which maximises reward
Consider all chosen Links L_{j1} ...
<=
Apply Constraints
Fix bandwidths
L
F
update link state
Iterate next flow

References

1. Shirmarz, A., Ghaffari, A. Automatic Software Defined Network (SDN) Performance Management Using TOPSIS Decision-Making Algorithm. J Grid Computing 19, 16 (2021). https://doi.org/10.1007/s10723-021-09557-z

2. H. Yao, X. Yuan, P. Zhang, J. Wang, C. Jiang and M. Guizani, "Machine Learning Aided Load Balance Routing Scheme Considering Queue Utilization," in IEEE Transactions on Vehicular Technology, vol. 68, no. 8, pp. 7987-7999, Aug. 2019, doi: 10.1109/TVT.2019.2921792

3. Huang, L., Ye, M., Xue, X. et al. Intelligent routing method based on Dueling DQN reinforcement learning and network traffic state prediction in SDN. Wireless Netw (2022). https://doi.org/10.1007/s11276-022-03066-x

4. Mohammadi, R., Javidan, R. EFSUTE: a novel efficient and survivable traffic engineering for software defined networks. J Reliable Intell Environ 8, 247–260 (2022). https://doi.org/10.1007/s40860-021-00139-0

5. BinSahaq, A., Sheltami, T., Mahmoud, A. et al. Fast and efficient algorithm for delay-sensitive QoS provisioning in SDN networks. Wireless Netw (2022). https://doi.org/10.1007/s11276-022-03028-3

6. Zhang, Y., Chen, M. Performance evaluation of Software-Defined Network (SDN) controllers using Dijkstra's algorithm. Wireless Netw (2022). https://doi.org/10.1007/s11276-022-03044-3

7. A. Santos Da Silva, C. C. Machado, R. V. Bisol, L. Z. Granville and A. Schaeffer-Filho, "Identification and Selection of Flow Features for Accurate Traffic Classification in SDN," 2015 IEEE 14th International Symposium on Network Computing and Applications, 2015, pp. 134-141, doi: 10.1109/NCA.2015.12.

8. D. Jayasinghe, W. H. Rankothge, N. D. U. Gamage, T. C. T. Gamage, S. D. L. S. Uwanpriya and D. A. H. M. Amarasinghe, "Network Traffic Prediction for a Software Defined Network Based Virtualized Security Functions Platform," 2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2021, pp. 1083-1088, doi: 10.1109/IEMCON53756.2021.9623169.

9. https://opennetworking.org/sdn-definition/