Link layer switches – Question 1

i.

|  |  |  |
| --- | --- | --- |
| Switch | MAC address | interface |
| B1 | A | 2 |
| B2 | A | 1 |
| E | 2 |
| B3 | A | 2 |
| E | 1 |
| B4 | A | 2 |
| E | 3 |

ii.

let us draw the APR table after a message was sent from A to E in the new network configuration (the only change is marked in red color):

|  |  |  |
| --- | --- | --- |
| Switch | MAC address | interface |
| B1 | A | 2 |
| B2 | A | 1 |
| E | 2 |
| B3 | **A** | **4** |
| E | 1 |
| B4 | A | 2 |
| E | 3 |

Note: the notation **Bi:j** stands for interface j in switch Bi.

* **Message from D to A**

let us write the message's path:

D 🡪 B4:1 🡪B4:2 🡪B3:1🡪B3:4🡪A

Hence, **this message gets to its destination**.

* **Message from C to A**

let us write the message's path:

C 🡪 B1:3 🡪 B1:2 🡪 B2:3 🡪 B2:1 🡪 no host connected…

Hence, **this message does not get to its destination**. That happens because switch B2 has not learned yet about A's new interface (in its table it is still connected via interface 1, when it is actually connected via interface 2).

Bonus question:

Let be the CDMA codes of users 1 and 2, where each code is a vector containing M bits (encoding 1 real bit). We also assume that the norm of each vector equals to M.

Let us define the inner-product that we will use for 2 arbitrary vectors :

We assume that the codes are orthonormal, hence:

* for every :
* For every :

When user k sends in time i some bit , it is encoded as a vector (remember that is a scalar and is a vector). When both users send a bit together we add their vectors:

Hence, the following holds:

Transition (1) is derived from the inner-product definition. Transition (2) holds due to the linearity property of an inner product. Transition (3) holds due to the orthonormality property. □