# Answer for Question 1

The precedence graphs for the schedules are shown below:

T2

T1

T3

S1:

T3

T2

T1

S2:

T2

T1

T3

S3:

T2

T1

T3

S4:

1. Since the precedence graphs are different from each other, none of the schedules are conflict equivalent.
2. The precedence graphs for the schedules S3 and S4 are serializable.
   1. S3 is conflict equivalent to the serial schedule:

R2(z),W2(x),W2(y),R3(x),R3(y),R3(z),R1(x),W1(x)

* 1. S4 is conflict equivalent to the serial schedule:

R2(z),W2(x),W2(y),W1(x),R1(x),R3(x),R3(y),R3(z)

1. S3 and S4 are not globally serializable. For global transactions (i.e., global schedule) to be serializable, two conditions are necessary:
   1. Each local schedule should be serializable.
   2. All sub-transactions of global transactions appear in the same order in the equivalent serial schedule at ALL sites.

In this case, condition b) is not met because for S3 the sub-transactions appear in the order T2🡪T3🡪T1 which is different to the order T2🡪T1🡪T3 for S4.

# Answer for Question 2

1. The wait-for graph is shown below. There are 3 deadlocks:
   1. T2🡪T7🡪T2
   2. T6🡪T1🡪T6
   3. T8🡪T4🡪T3🡪T8

Site B

Site F

b(T7)

T2

b(T2)

T7

T5

Site C

Site E

c(T1)

T6

e(T6)

T1

Site A

Site D

a(T4)

T8

d(T3)

T4

Site G

Site H

g(T8)

T3

h(T6)

1. The best organization of the sites for this scenario is shown below.

DDBF

B

F

C

E

A

D

G

H

DDCE

DDAD

DDGH

DDBFCE

DDADGH

DDBFCEADGH

This organization of the sites is the best one because it minimizes the communication cost for deadlock detection.

* Deadlock 1) can be detected by DDBF
* Deadlock 2) can be detected by DDCE
* Deadlock 3) can be detected by DDADGH

# Answer for Question 3

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Communication Topology** | **Number of Messages** | **Number of Rounds** |
|  |  |  |  |
|  | Centralized 2PC | N×4 | 2 |
|  | Linear 2PC | N×2 | N |
|  | Distributed 2PC | (N+1)×N | 1 |
|  |  |  |  |
|  | Centralized 3PC | N×5 | 2.5 |

1. In 3PC, it is possible for a node to be in PRECOMIT state and eventually be aborted. Such situation is described below:
   1. In phase 2 of 3PC, the coordinator collects votes which are all YES.
   2. After it successfully sends PRECOMMIT message to the first participant, both the coordinator and the first participant fail.
   3. All other operational sites start the termination protocol by electing a new coordinator.
   4. Elected coordinator collects states from all other sites.
   5. It turns out that all sites report their states as **uncertain**, so the coordinator decides **abort**.
   6. After the decision is made, the original coordinator and first participant recover from failure.
   7. According to the recovery protocol, they follow the global **abort** decision and change from PRECOMMIT state to **abort** state.

# Answer for Question 4

1. Distribution of tuples on the value of attribute Y for relations R & S is shown below:

Assume we use the same partition vector (K0) for the range partitioning of R and S on attribute Y, i.e., the fragments of on the nodes are:

* 1. R1 = σY≤ K0(R)
  2. R2 = σY> K0(R)
  3. S1 = σY≤ K0(S)
  4. S2 = σY> K0(S)

In this way, equal-join between R and S can be executed in parallel:

* nodes 1) & 3)🡪R1 join S1
* nodes 2) & 4)🡪R2 join S2

So the communication cost will be MIN(|R1|, |S1|) + MIN(|R2|, |S2|). The relationship between communication cost and K0 is shown in the diagram below from which we conclude that as long as K0 is in the range of (4000, 12000), the communication cost will be at the minimum value of 32000.

1. Based on the distribution of tuples on the value of attribute Y for relations R & S, we can calculate the number of join result expected for node 1 & 3 or 2 & 4. The result is shown in the diagram below. From the diagrams above and below, we conclude that range partitioning with K0=8000 gives the best balance while minimizing the communication cost.
2. The number of join comparisons during hash-join for
   * nodes 1) & 3) = |R1| × |S1| / 1000
   * nodes 2) & 4) = |R2| × |S2| / 1000

The calculation result is show in the diagram below. For range partitioning with K0=8000, the number of join comparisons during hash-join for nodes 1) & 3) is 256000, while for nodes 2) & 4) is 576000.