Distributed Evaluation of Enhanced Path Queries

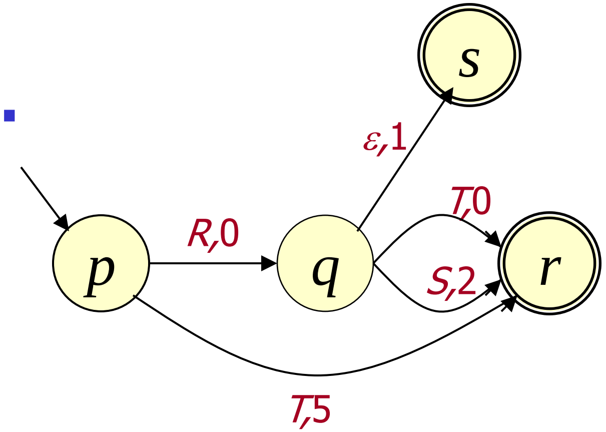
Dan C. Stefanescu, Alex Thomo

Label setting vs. Label Correcting Algorithms

- Label setting algorithms: Use priority queue
  - E.g. Dijkstra's
  - When a node is labeled we know for sure that it's the best label, i.e. it will be the weight of the shortest path from the source to that node.

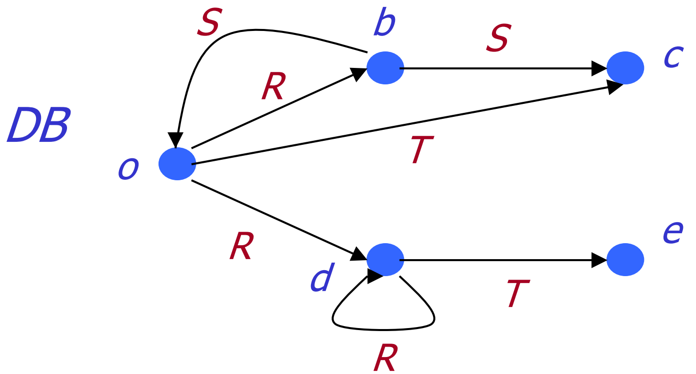
- Label correcting algorithms: Use FIFO queue
  - E.g. Pallotino's
  - When a node is labeled we don't know for sure that it's the best label, possibly we will improve it later.
  - Appropriate to extend to distributed setting…

Let's apply it…



We will build the green graph implicitly and lazily.

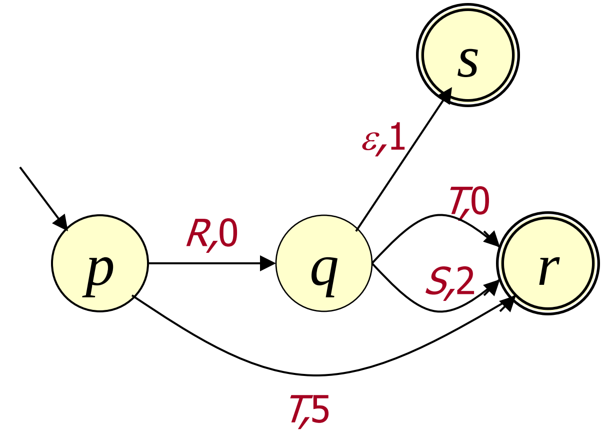
Along the way we will apply the shortest path general algorithm.



Queue : (o,p,0)

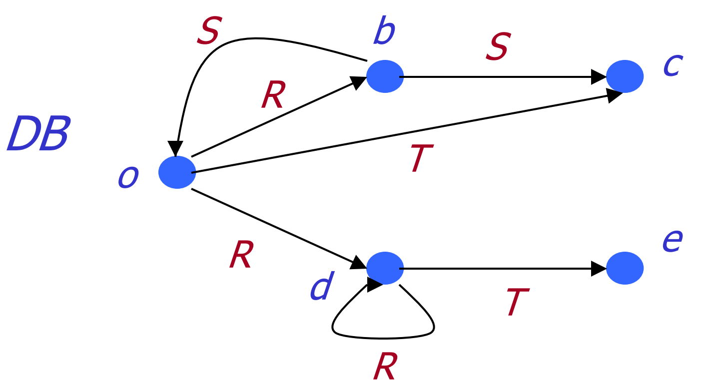
Object-State-Weight Table : Here we store the "labeling"s of the green graph nodes.

Let's apply it…



We will build the green graph implicitly and lazily.

Along the way we will apply the shortest path general algorithm.

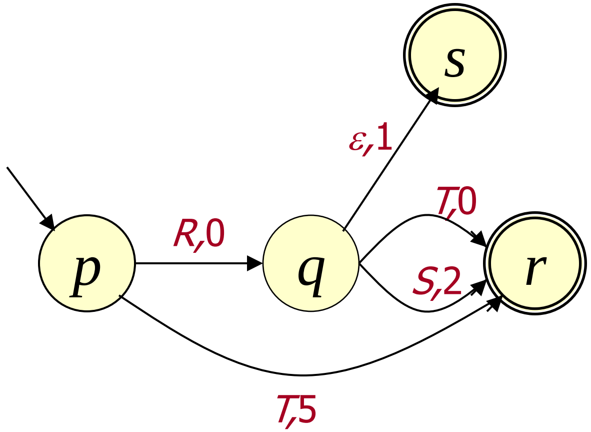


Queue : (c,r,5) (b,q,0) (d,q,0)

Object-State-Weight Table : Here we store the "labeling"s of the green graph nodes.

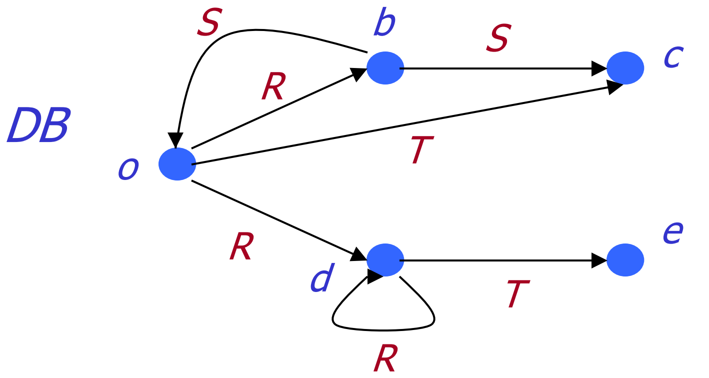
(o,p,0)

Let's apply it…



We will build the green graph implicitly and lazily.

Along the way we will apply the shortest path general algorithm.

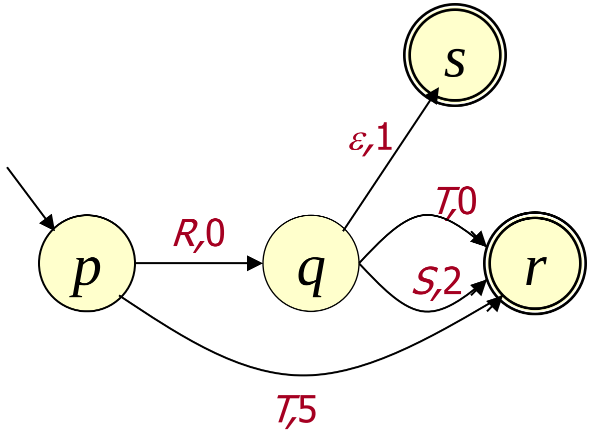


Queue : (b,q,0) (d,q,0)

Object-State-Weight Table : Here we store the "labeling"s of the green graph nodes.

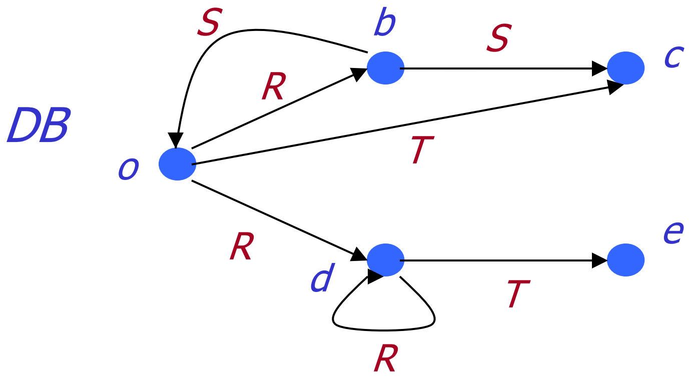
(o,p,0) (c,r,5)

Let's apply it…



We will build the green graph implicitly and lazily.

Along the way we will apply the shortest path general algorithm.

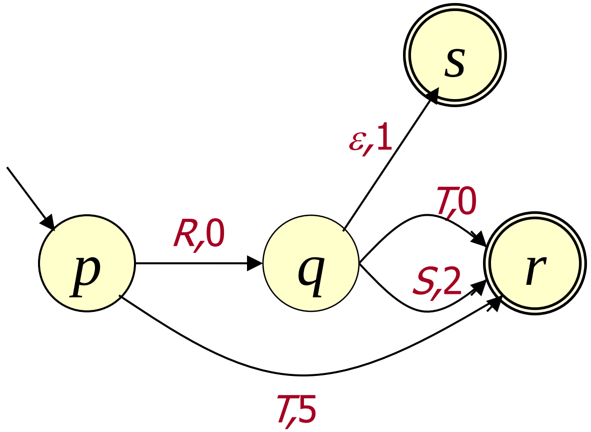


Queue : (d,q,0) (b,s,1) (c,r,2)

Object-State-Weight Table : Here we store the "labeling"s of the green graph nodes.

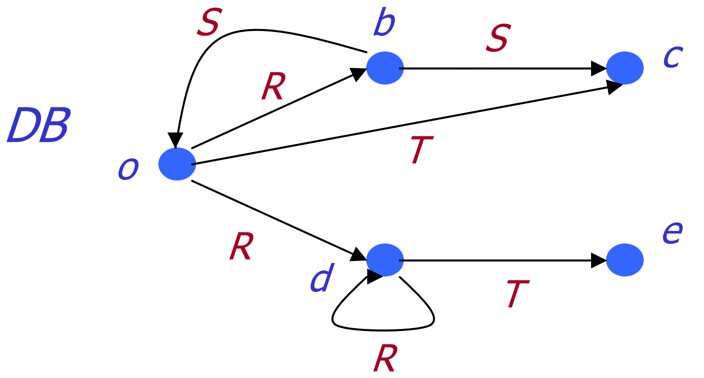
(o,p,0) (c,r,5) (b,q,0)

Let's apply it…



We will build the green graph implicitly and lazily.

Along the way we will apply the shortest path general algorithm.

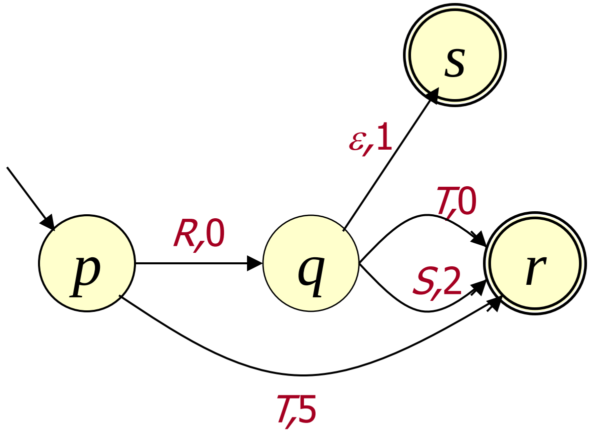


Queue : (b,s,1) (c,r,2) (d,s,1) (e,r,0)

Object-State-Weight Table : Here we store the "labeling"s of the green graph nodes.

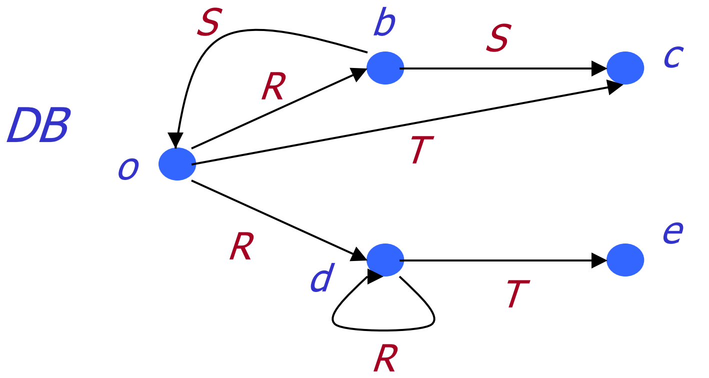
(o,p,0) (c,r,5) (b,q,0) (d,q,0)

Let’s apply it…



We will build the green graph implicitly and lazily.

Along the way we will apply the shortest path general algorithm.

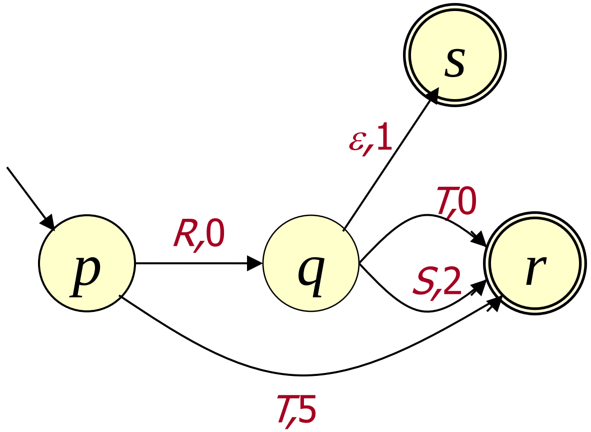


Queue : (c,r,2) (d,s,1) (e,r,0)

Object-State-Weight Table : Here we store the “labeling”s of the green graph nodes.

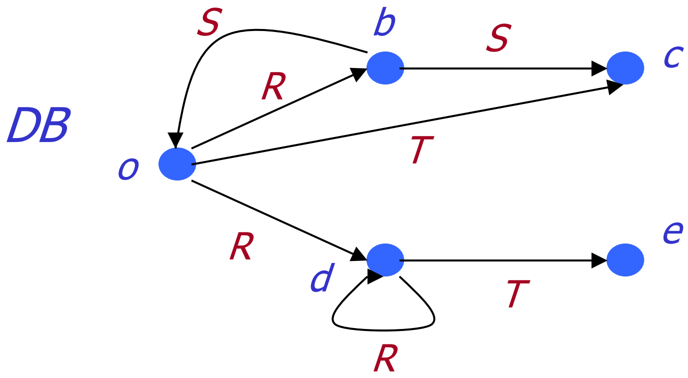
(o,p,0) (c,r,5) (b,q,0) (d,q,0) (b,s,1)

Let's apply it…



We will build the green graph implicitly and lazily.

Along the way we will apply the shortest path general algorithm.



Queue : (d,s,1) (e,r,0)

Object-State-Weight Table : Here we store the "labeling"s of the green graph nodes.

(o,p,0) (c,r,2) (b,q,0) (d,q,0) (b,s,1)

…

Distributed Algorithm

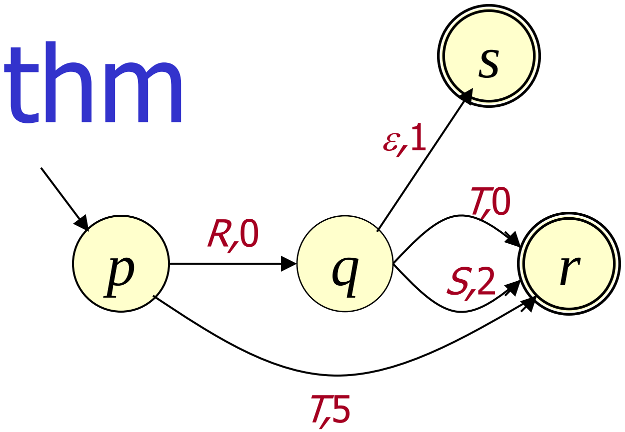
- The nodes are partitioned in different processors, which don't share memory.
- Communication is achieved through asynchronous message passing.

Idea

- Partition the *Object-State-Weight* table among the processors
- When dequeueing see whether the corresponding object is in the local *Object-State-Weight* table.
  - If yes, proceed as previously
  - Otherwise pack the dequeued triple in a message and send it to the processor holding the real DB node.

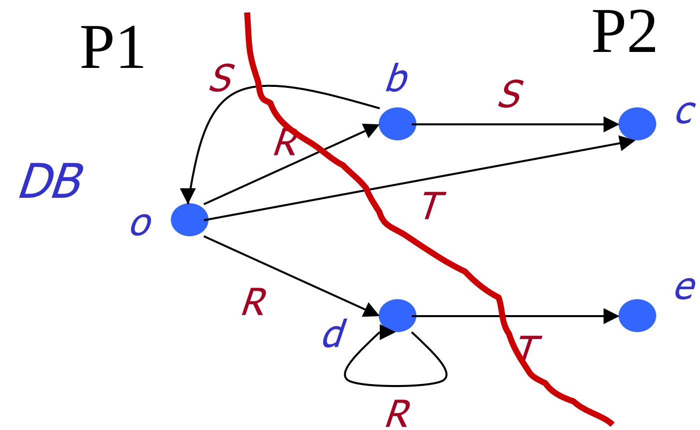
- Terminate when the processing queues of all processors are empty and there is no message sent but not yet received.

Distributed Algorithm



Suppose:

- o, d are in processor P1
- b, c, e are in processor P2



P1:

Queue : (o,p,0)

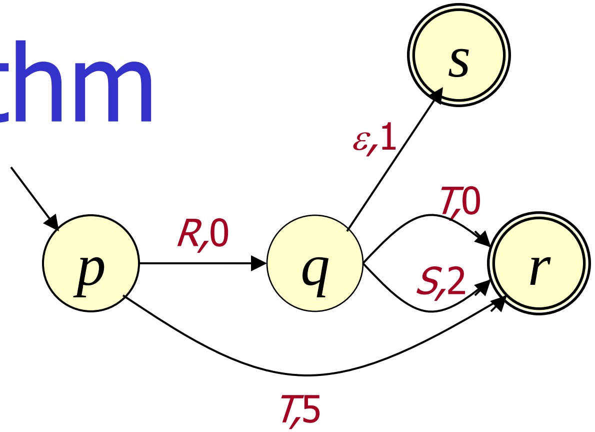
Object-State-Weight Table :

P2:

Queue :

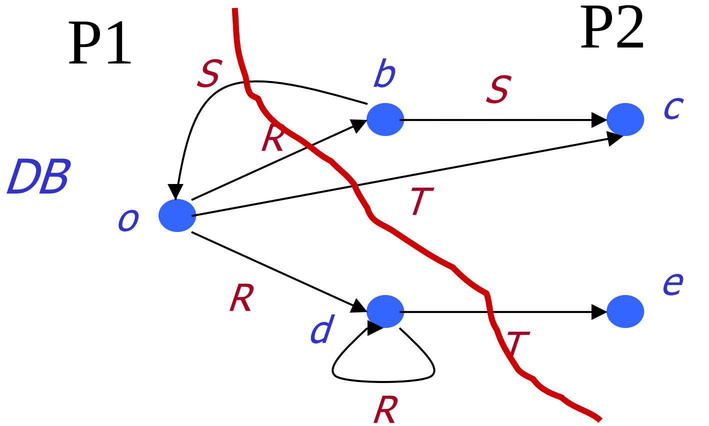
Object-State-Weight Table :

Distributed Algorithm



Suppose:

- o, d are in processor P1
- b, c, e are in processor P2



P1:

Queue : (c,r,5) (b,q,0) (d,q,0)

Object-State-Weight Table :

(o,p,0)

message (c,r,5)

P2:

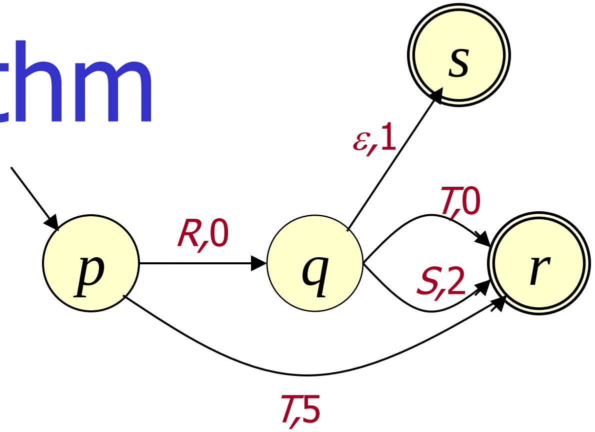
Queue :

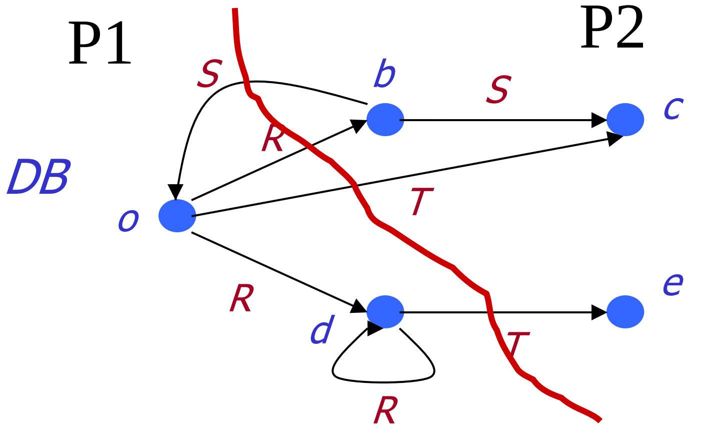
Object-State-Weight Table :

Distributed Algorithm

Suppose:

- o, d are in processor P1
- b, c, e are in processor P2





P1:

Queue : (b,q,0) (d,q,0)

Object-State-Weight Table :

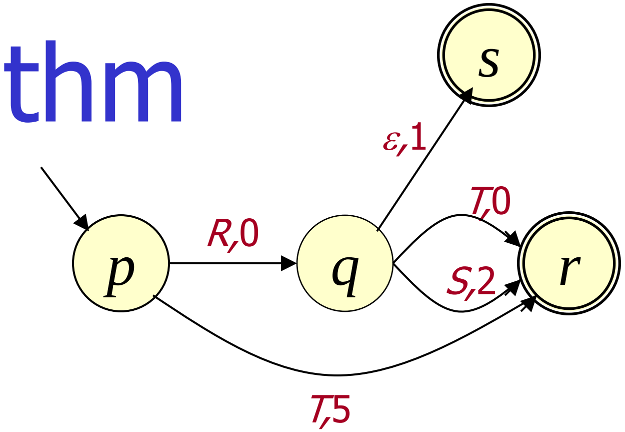
(o,p,0)

P2:

Queue : (c,r,5)

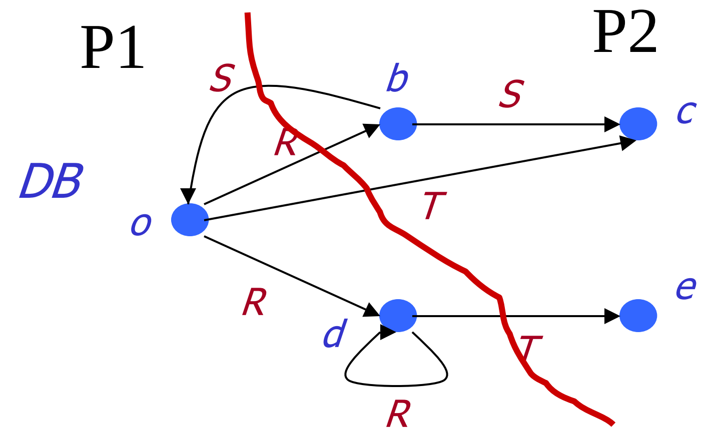
Object-State-Weight Table :

Distributed Algorithm



Suppose:

- o, d are in processor P1
- b, c, e are in processor P2



P1:

Queue : (b,q,0) (d,q,0)

Object-State-Weight Table :

(o,p,0)

message (b,q,0)

P2:

Queue :

Object-State-Weight Table :

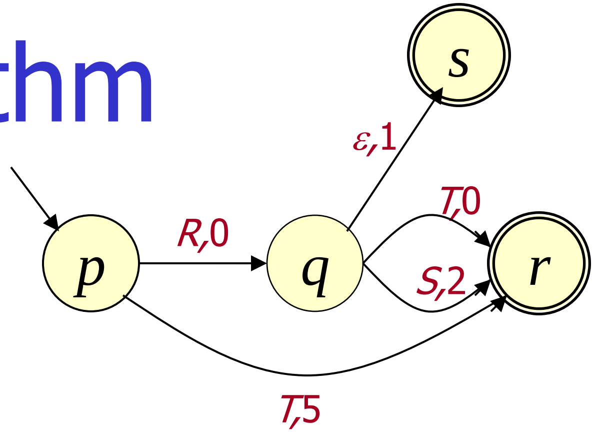
(c,r,5)

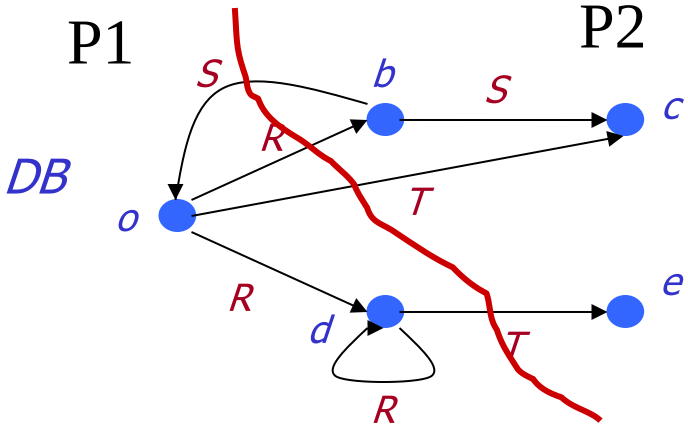
Distributed Algorithm

Suppose:

o, d are in processor P1

b, c, e are in processor P2





P1:

Queue : (d,q,0)

Object-State-Weight Table :

(o,p,0)

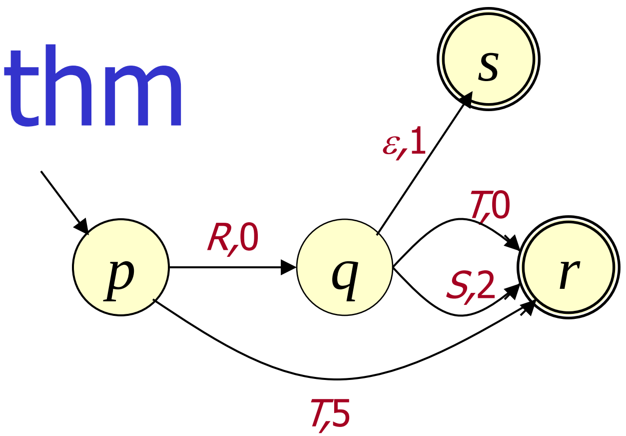
P2:

Queue : (b,q,0)

Object-State-Weight Table :

(c,r,5)

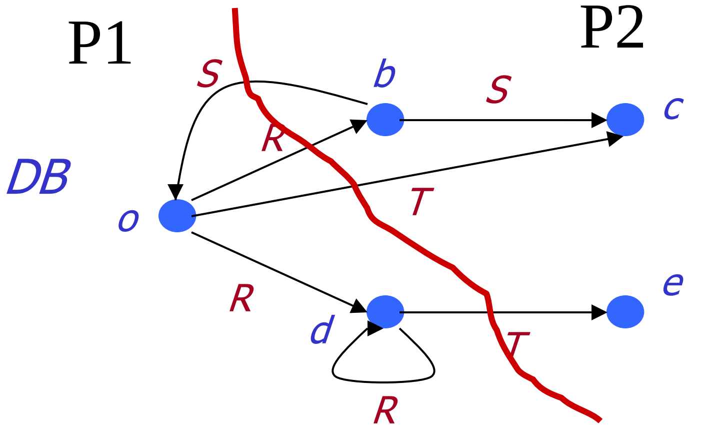
Distributed Algorithm



Suppose:

o, d are in processor P1

b, c, e are in processor P2



P1:

Queue : (d,q,0)

Object-State-Weight Table :

(o,p,0)

P2:

Queue : (b,s,1) (c,r,2) (o,r,2)

Object-State-Weight Table :

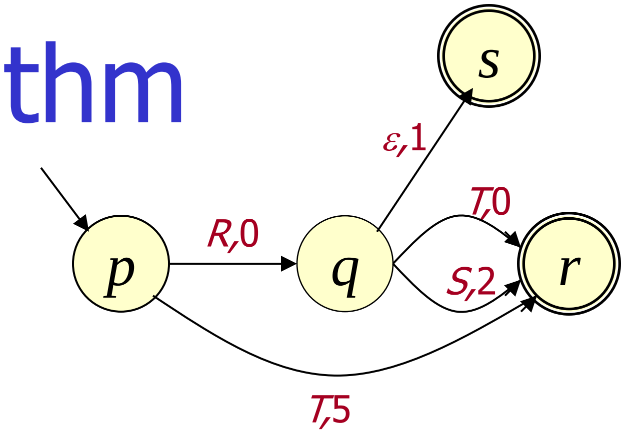
(c,r,5) (b,q,0)

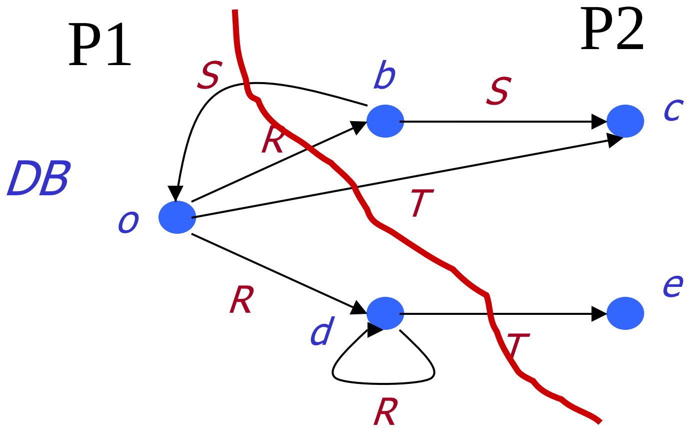
Distributed Algorithm

Suppose:

o, d are in processor P1

b, c, e are in processor P2





P1:

Queue : (d,q,0)

Object-State-Weight Table :

(o,p,0)

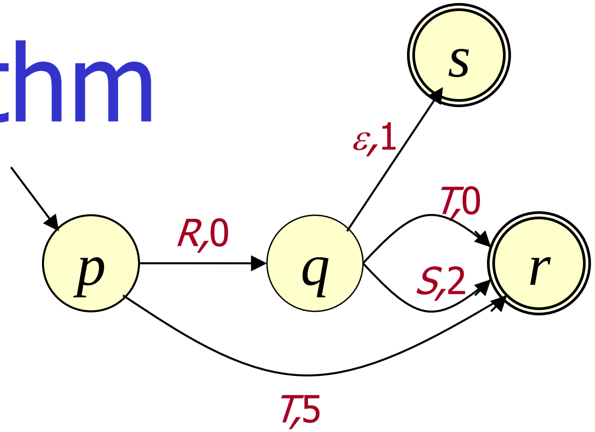
P2:

Queue : (c,r,2) (o,r,2)

Object-State-Weight Table :

(c,r,5) (b,q,0) (b,s,1)

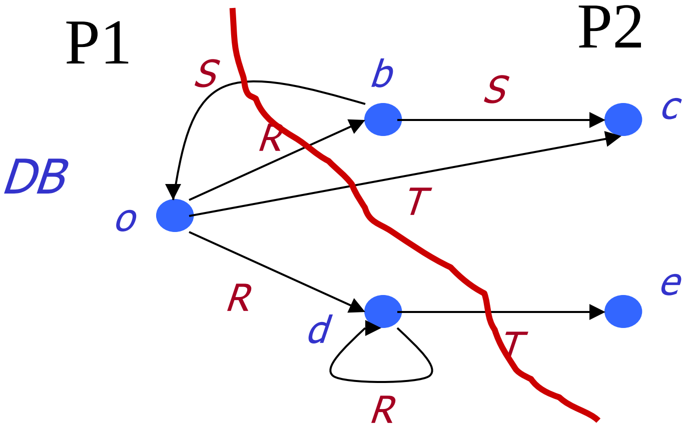
Distributed Algorithm



Suppose:

o, d are in processor P1

b, c, e are in processor P2



P1:

Queue : (d,s,1) (e,r,0)

Object-State-Weight Table :

(o,p,0) (d,q,0)

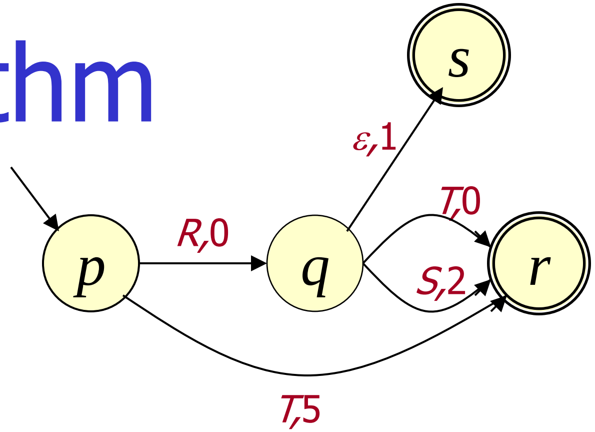
P2:

Queue : (o,r,2)

Object-State-Weight Table :

(c,r,2) (b,q,0) (b,s,1)

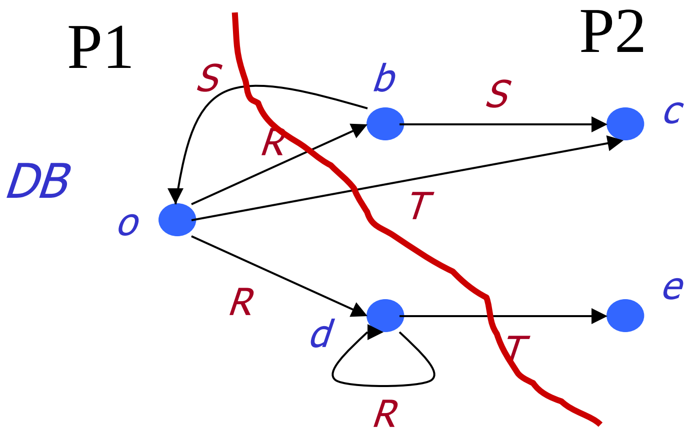
Distributed Algorithm



Suppose:

o, d are in processor P1

b, c, e are in processor P2



P1:

Queue : (e,r,0)

Object-State-Weight Table :

(o,p,0) (d,q,0) (d,s,1)

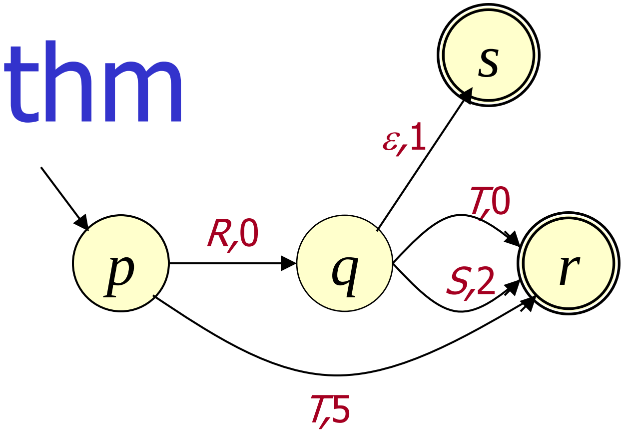
P2:

Queue : (o,r,2)

Object-State-Weight Table :

(c,r,2) (b,q,0) (b,s,1)

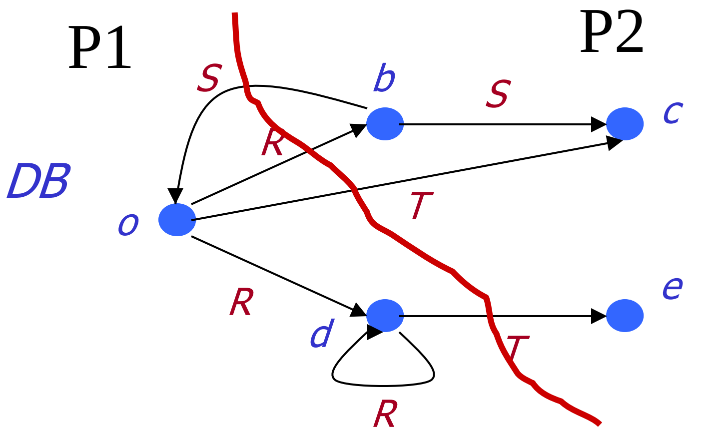
Distributed Algorithm



Suppose:

o, d are in processor P1

b, c, e are in processor P2



P1:

Queue : (e,r,0)

Object-State-Weight Table :

(o,p,0) (d,q,0) (d,s,1)

message (o,r,2)

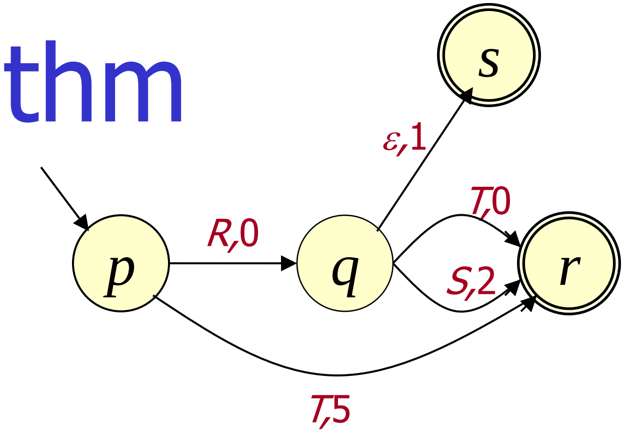
P2:

Queue : (o,r,2)

Object-State-Weight Table :

(c,r,2) (b,q,0) (b,s,1)

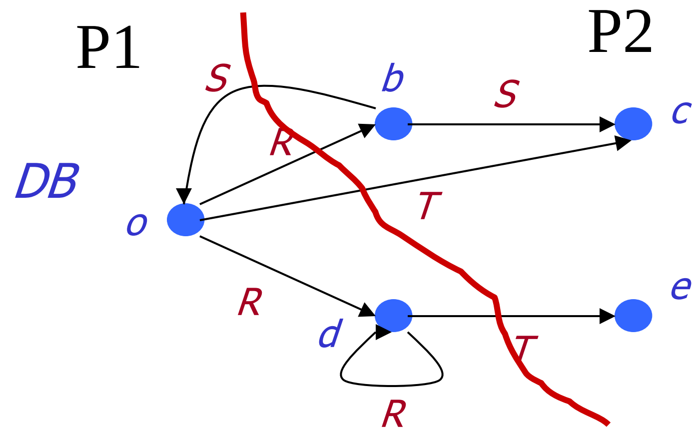
Distributed Algorithm



Suppose:

o, d are in processor P1

b, c, e are in processor P2



P1:

Queue : (o,r,2) (e,r,0)

Object-State-Weight Table :

(o,p,0) (d,q,0) (d,s,1)

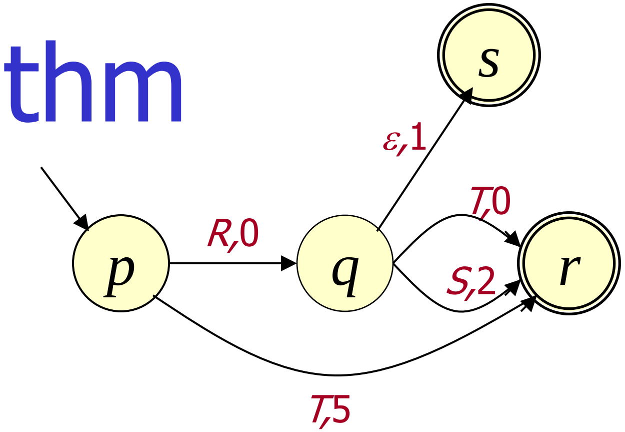
P2:

Queue :

Object-State-Weight Table :

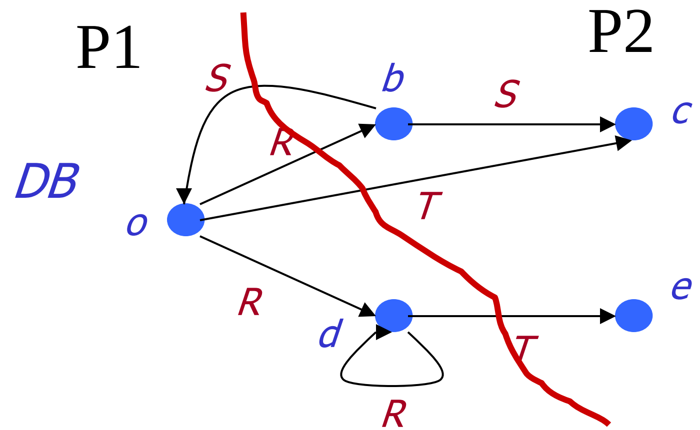
(c,r,2) (b,q,0) (b,s,1)

Distributed Algorithm



Suppose:

- o, d are in processor P1
- b, c, e are in processor P2



P1:

Queue : (e,r,0)

Object-State-Weight Table :

(o,p,0) (d,q,0) (d,s,1) (o,r,2)

message (e,r,0)

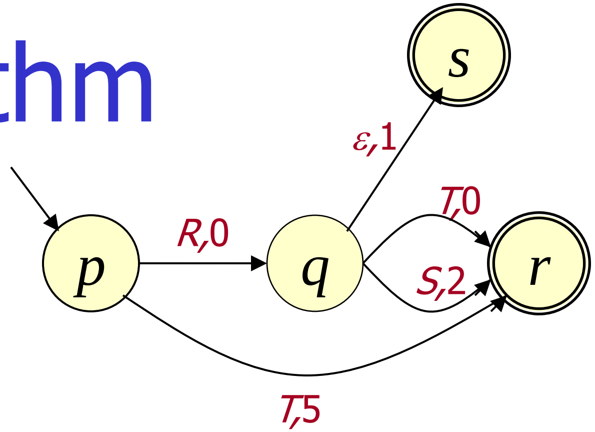
P2:

Queue :

Object-State-Weight Table :

(c,r,2) (b,q,0) (b,s,1)

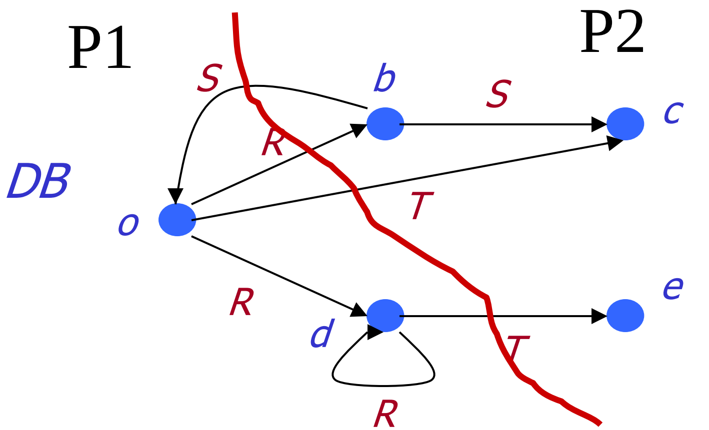
Distributed Algorithm



Suppose:

o, d are in processor P1

b, c, e are in processor P2



P1:

Queue :

Object-State-Weight Table :

(o,p,0) (d,q,0) (d,s,1) (o,r,2)

P2:

Queue : (e,r,0)

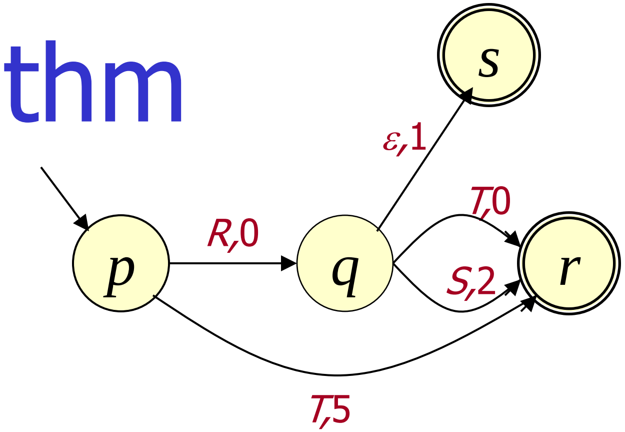
Object-State-Weight Table :

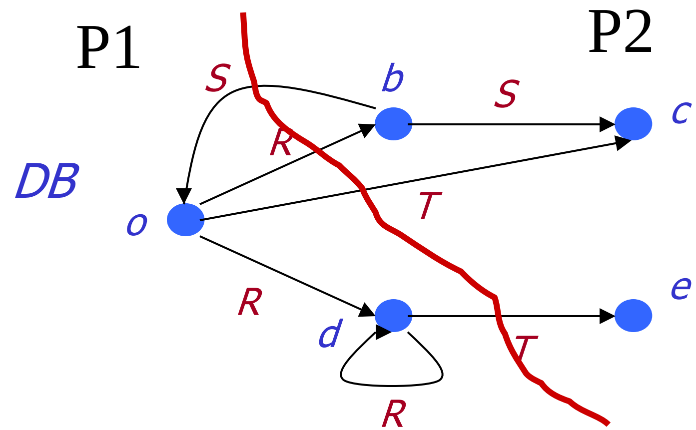
(c,r,2) (b,q,0) (b,s,1)

Distributed Algorithm

Suppose:

- o, d are in processor P1
- b, c, e are in processor P2





P1:

Queue :

Object-State-Weight Table :

(o,p,0) (d,q,0) (d,s,1) (o,r,2)

P2:

Queue :

Object-State-Weight Table :

(c,r,2) (b,q,0) (b,s,1) (e,r,0)

References

- Dan C. Stefanescu, Alex Thomo. Enhanced Regular Path Queries on Semistructured Databases. EDBT Workshops 2006: 700-711
- Dan C. Stefanescu, Alex Thomo, Lida Thomo. Distributed evaluation of generalized path queries. SAC 2005: 610-616
- Gösta Grahne, Alex Thomo. Query Answering and Containment for Regular Path Queries under Distortions. FoIKS 2004: 98-115