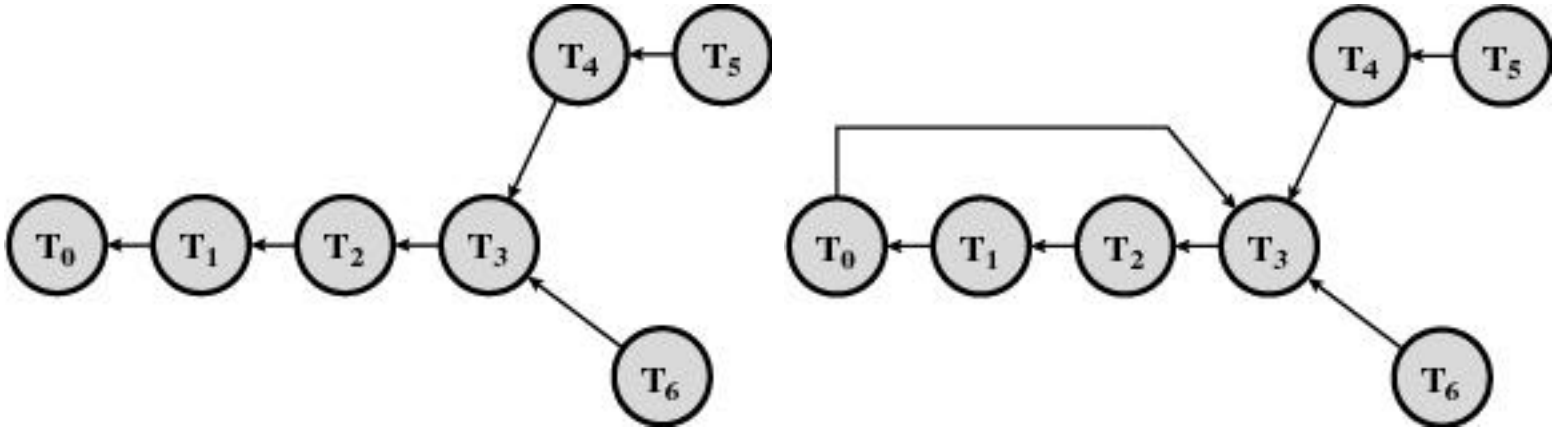
```
/* Data object Dj receiving a lock_request(Ti) */
    if (Locked_by(Dj) == null)
        send(granted);
    else
    {
        send not granted to Ti;
        send Locked_by(Dj) to Ti
    }


/* Transaction Ti makes a lock request for data object Dj */
    send lock_request(Ti) to Dj;
    wait for granted/not granted;
    if (granted)
    {
        Locked_by(Dj) = Ti;
        Held_by(Ti) = φ;
    }
    else /* suppose Dj is being used by transaction Tj */
    {
        Held_by(Ti) = Tj;
        Enqueue(Ti, Request_Q(Tj));
        if (Wait_for(Tj) == null)
            Wait_for(Ti) = Tj ;
        else
            Wait_for(Ti) = Wait_for(Tj);
        update(Wait_for(Ti), Request_Q(Ti));
    }
```

```
/* Transaction Tj receiving an update message */

    if (Wait_for(Tj) != Wait_for(Ti))
        Wait_for(Tj) = Wait_for(Ti);
    if (intersect(Wait_for(Tj), Request_Q(Tj)) = null)
        update(Wait_for(Ti), Request_Q(Tj);
    else
    {
            DECLARE DEADLOCK;
            /* initiate deadlock resolution as follows */
            /* Tj is chosen as the transaction to be aborted */
            /* Tj releases all the data objects it holds */
            send_clear(Tj, Held_by(Tj));
            allocate each data object Di held by Tj to the first
                requester Tk in Request_Q(Tj);
            for (every transaction Tn in Request_Q(Tj) requesting
                data object Di held by Tj)
            {
                Enqueue(Tn, Request_Q(Tk));
            }
    }


/* Transaction Tk receiving a clear(Tj, Tk) message */
    purge the tuple having Tj as the requesting transaction from
    Request_Q(Tk);
```

Figure 14.14 A Distributed Deadlock Detection Algorithm



Transaction	Wait_for	Held_by	Request_Q		Transaction	Wait_for	Held_by	Request_Q
T_0	nil	nil	T_1		T_0	T_0	T_3	T_1
T_1	T_0	T_0	T_2		T_1	T_0	T_0	T_2
T_2	T_0	T_1	T_3		T_2	T_0	T_1	T_3
T_3	T_0	T_2	T_4, T_6		T_3	T_0	T_2	T_4, T_6, T_0
T_4	T_0	T_3	T_5		T_4	T_0	T_3	T_5
T_5	T_0	T_4	nil		T_5	T_0	T_4	nil
T_6	T_0	T_3	nil		T_6	T_0	T_3	nil

(a) State of system before request **(b) State of system after T_0 makes a request to T_3**

Figure 14.15 Example of Distributed Deadlock Detection Algorithm of Figure 14.14