Termination of distributed calculation

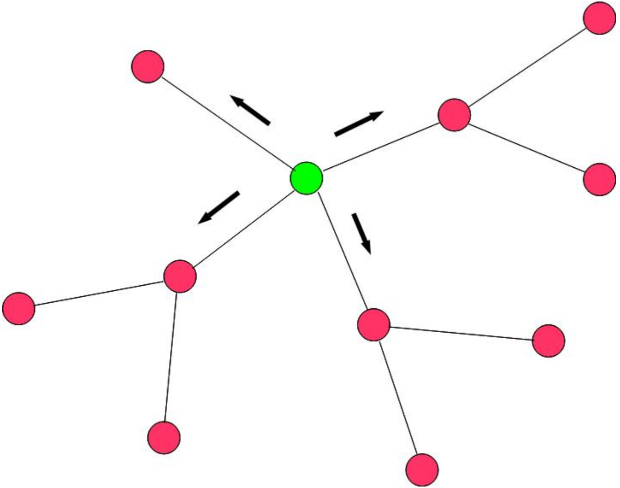
Algoritms

Algorithms, which are detecting termination of the computation, are closely related (similar) to algorithms, which are detecting deadlocks.
In deadlock algorithms we are looking for processes that remain themselves in the passive state, in other processes the execution can continue.
In termination algorithms, we are interested in the fact that all processes are in the passive state which can result from termination of the process or from the deadlock.

Similarly as in previous lectures, we can find algorithms, which are based on different principles.
As examples we present the test of diffusion algorithm, and the test serving as an auxiliary (additional) algorithm.

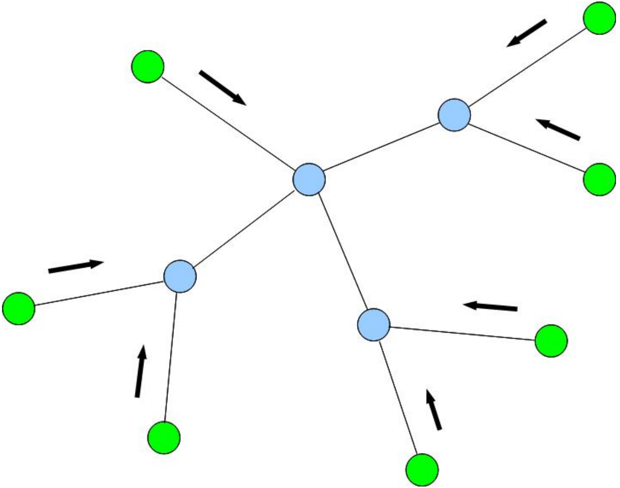
Termination of distributed calculation

Dijkstra-Scholten



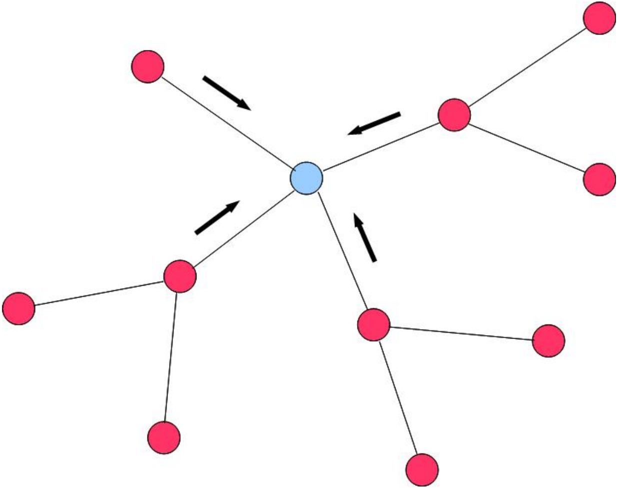
Termination of distributed calculation

Dijkstra-Scholten



Termination of distributed calculation

Dijkstra-Scholten



Termination of distributed calculation

Dijkstra-Scholten

```
receiving MESSAGE from j do                 { příjem žádosti aplikace }
  begin
    if DefIn=0
    then Parent := j
    else Others := Others+j;
    DefIn := DefIn+1
  end

receiving SIGNAL from j do                  { příjem odpovědi aplikace }
  DefOut:=DefOut-1;
```

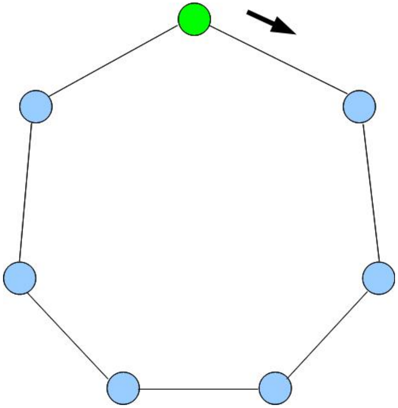
Termination of distributed calculation

Dijkstra-Scholten

```
sending MESSAGE to j do                          { odeslání žádosti aplikace }
   { possible if DefIn>0 }
   DefOut := DefOut+1;

sending SIGNAL to (Oth=any of Others) do  { odeslání odpovědi aplikace }
   { possible if (DefIn>1 }
   begin
      Others := Others-Oth;
      DefIn := DefIn-1
   end

sending SIGNAL to Parent do                      { odeslání odpovědi aplikace }
   { possible if (DefIn=1 and DefOut=0) }
   DefIn := DefIn-1

begin                            { inicializace }
   Defin:=0; Defout:=0; Others:=
end
```

Termination of distributed calculation

Dijkstra-Feijen-Van Gasteren



Termination of distributed calculation

Dijkstra-Feijen-Van Gasteren

```
receiving MESSAGE do                          { příjem zprávy aplikace }
   State := ACTIVE

waiting MESSAGE or State=TERMINATED do
   State := PASSIVE                           { čekání na zprávu aplikace }

sending MESSAGE to j begin        { odeslání zprávy procesu s indexem j>i }
   if i<j then Color := BLACK

when received TOKEN(ct) from i+1 do           { příjem zprávy TOKEN }
   begin
      TPresent := T;
      TColor := ct;
      if i=0 then
         if Color=WHITE and TColor=WHITE
         then { TERMINATION DETECTED }
         else TColor := WHITE
   end
```

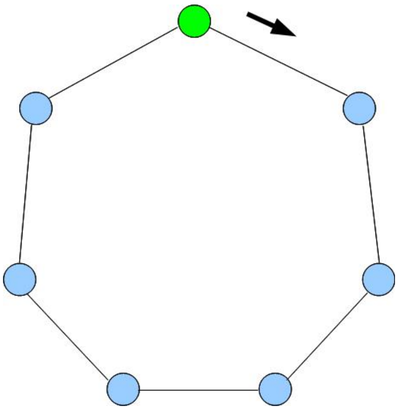
Termination of distributed calculation

Dijkstra-Feijen-Van Gasteren

```
when TPresent and State=PASSIVE do
   begin                                  { předání zprávy TOKEN následníkovi }
      if Color=BLACK then TColor := BLACK;
      TPresent := F;
      send TOKEN(TColor) to i-1;
      Color := WHITE
   end

begin                                     { inicializace }
   TPresent := F; Color := WHITE
end
```

Termination of distributed calculation

Misra



Termination of distributed calculation

Misra

```
when received MESSAGE do            { příjem zprávy aplikace }
    begin
        State := ACTIVE;
        Color := BLACK
    end

when waiting MESSAGE do             { čekání na zprávu aplikace }
    State := PASSIVE

when received TOKEN(j) do           { příjem zprávy TOKEN }
    begin
        nb := j;
        TPresent := T;
        if nb=Size(C) and Color=WHITE then
            { TERMINATION DETECTED }
    end
```

Termination of distributed calculation

Misra

```
when TPresent and State=PASSIVE        { odeslání zprávy TOKEN }
   begin
      if Color=BLACK then nb := 0
      else nb := nb+1;
      send TOKEN(nb) to Succesor(C,i);
      Color := WHITE;
      TPresent := F
   end

begin                                   { inicializace }
   Color := BLACK; TPresent := F; nb := 0
end
```

Protection against "errors"of processes

For many applications it is sufficient, if the application steps arrange only a subset of processes, and failures of some of them will not affect the result. Such subsets can be *quora* .

- The simplest type of quorum is a subset of at least $(n+1)/2$ elements, such quorum is referred to as *Majority* .
- Maekawa's quorum has a number of elements equal to $\sqrt{n}$ for n processes, related *matrix quorum* formed by processes in the i-th column and j-th row of the matrix has $2\sqrt{(n)}-1$ elements.
- Another interesting class of quora is *tree quorum*. The quorum is a path between the root and a leaf of the binary tree whose nodes are processes. If any of the processes in this path is broken, it can be replaced by broken process „children" and routes over them to the leaves.

The quorum's mechanisms that allows to select a set of processes capable to perform calculation's steps are basis of mechanisms that allow to ensure the consistency of *replicated* data in any calculation process.

Quora

set of nodes

$$S = \{s_1, s_2, ...s_n\}$$

coterie

$$C = \{Q_1, Q_2, ...Q_m\},\ Q_i \neq \emptyset,\ Q_i \subseteq S$$

quorum

$$Q_i \cap Q_j \neq \emptyset,\ Q_i, Q_j \in C,\ i \neq j \quad \text{- intersection}$$

$$Q_i \not\subset Q_j, \quad Q_i, Q_j \in C,\ i \neq j \quad \text{- minimality}$$

Quora

Majority quorum

quorum size = (n+1)/2

Maekawa's quorum

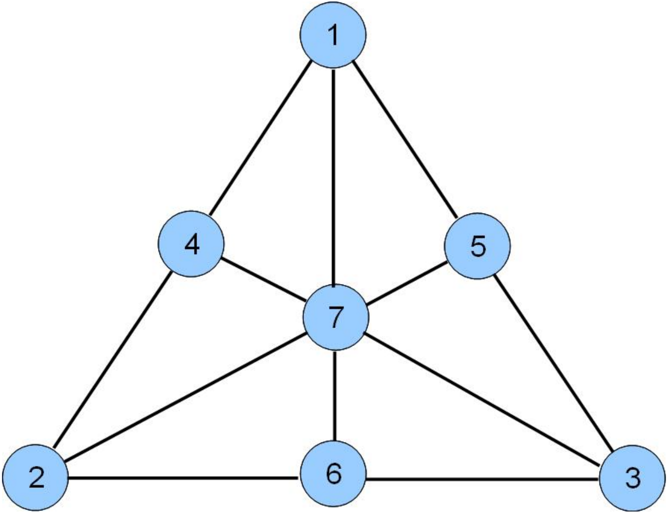
quorum size = √ n

Tree quorum

quorum size = √ n . . . (n+1)/2

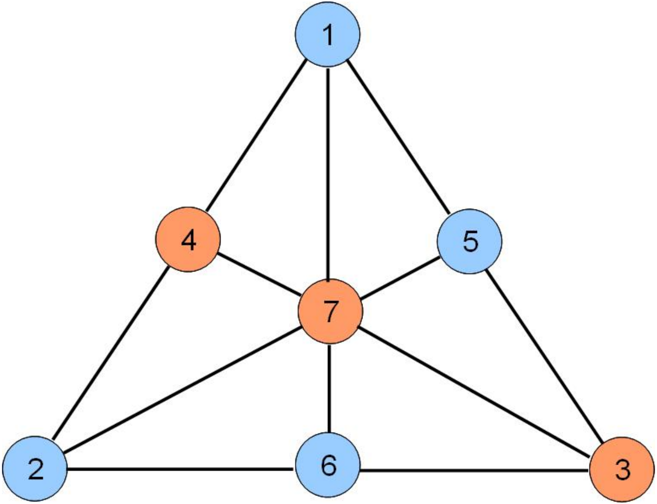
Quora

Maekawa



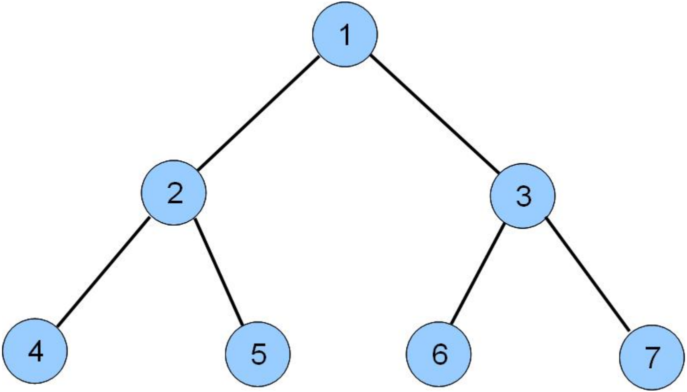
Quora

Maekawa



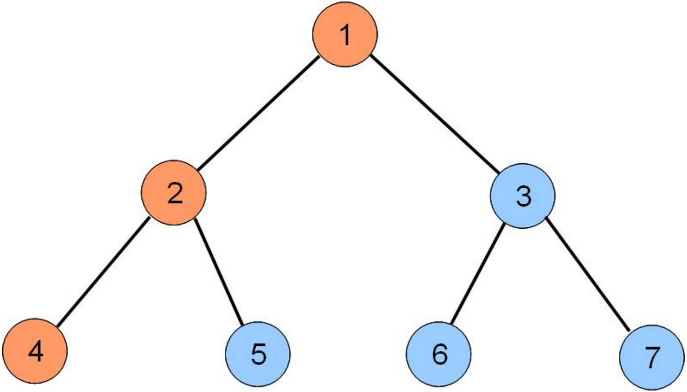
Quora

Tree quorum



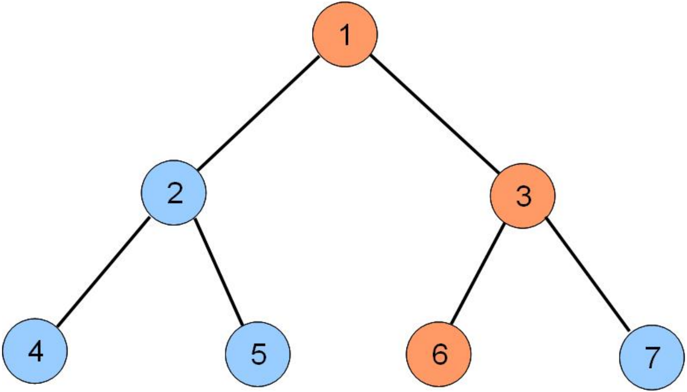
Quora

Tree quorum



Quora

Tree quorum



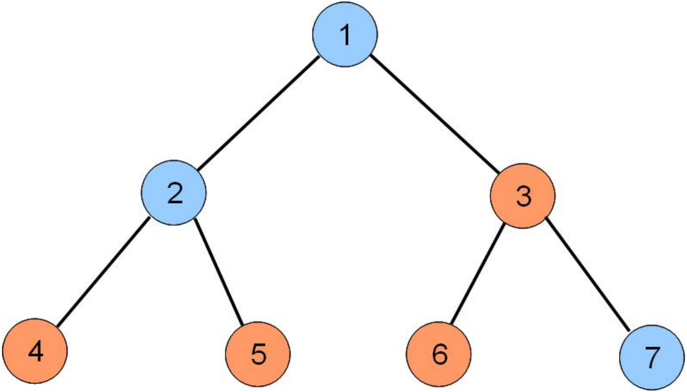
Quora

Tree quorum - Agrawal-El Abbadi

```
function GetQorum (Tree:tree) : quorumset;
var left, right : quorumset;
begin
   if Empty(Tree) then
     Return ({ });
   else
     if GrantsPermission(Tree^.Node) then
       return ({Tree^.Node} U GetQuorum(Tree^.LeftChild))
         or
       return ({Tree^.Node} U GetQuorum(Tree^.RightChild))
     else  (* ... handling Tree^.Node failure ... *)
       return (GetQuorum(Tree^.LeftChild) U GetQuorum(Tree^.RightChild))
end.
```

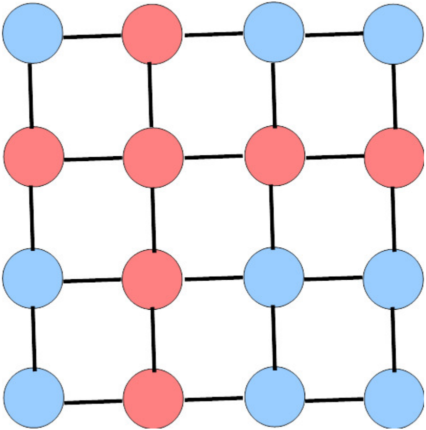
Quora

Tree quorum - Agrawal-El Abbadi



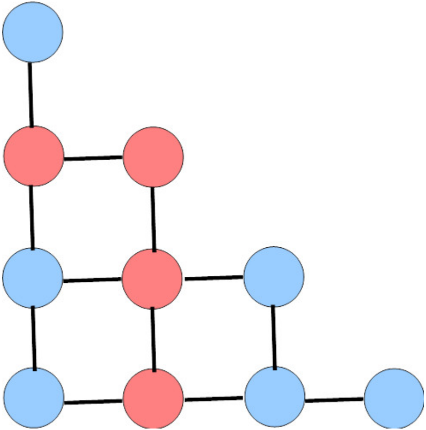
Quora

Square quorum



Quora

Triangular quorum



Quora

Quora usage