Mobile Ad hoc Networks

一种无基础设施的无线网络，由动态移动的节点组成

- 它是在发送方和接收方之间以最有效的方式传输消息的过程
- 节点是可移动的，意味着必须动态地计算路由，这就需要高效的路由算法
- 在MANET中保持连接也是另一个需要解决的问题
- 机器人网络是网络的另一个例子，在这样的网络中，机器人的协调操作需要保持网络随时连接。

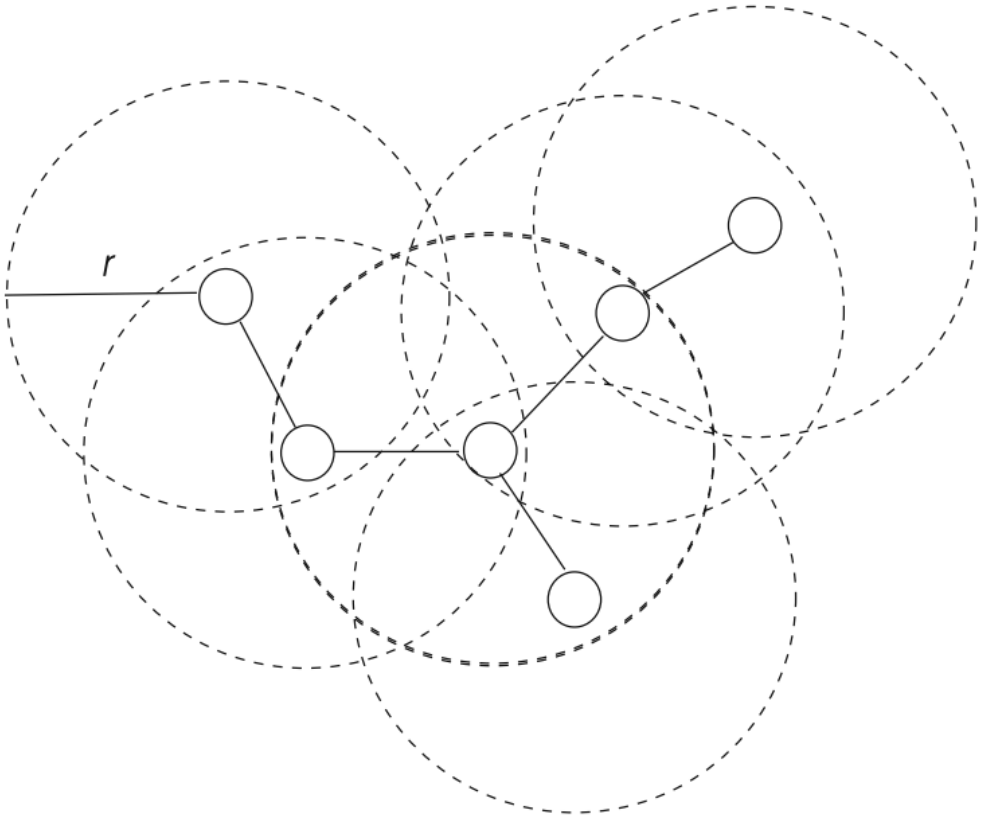
Wireless Sensor Networks

无线传感器网络(WSN)由传感器网络和无线收发器及控制器组成。

在无线传感器网络中，有效地利用网络协议将数据发送到接收端，并保持网络的连接是无线传感器网络要解决的主要问题

传感器网络大多是静止的，需要低功耗的操作，这比在manet中管理功耗更重要

通过图形可以方便地对MANET或WSN进行建模，将路由、连通性等问题转移到图域，利用图的方法进行求解。例如，利用求权图中两个节点之间的最短距离的方法可以解决有效的路由问题



- 表示WSN的图中的节点只能与它的邻居通信
- 该图显示一个具有节点的无线网络，可以在半径为r米的范围内发送和接收无线电信号
- 我们可以通过一条边将传输范围内的节点连接起来，得到如图所示的图形。

Models

我们可以将被广泛接受的分布式系统消息传递模型形式化定义为：

- 一个节点上的进程pi仅通过交换messages与其他进程通信 。
- 每个进程pi的状态si∈S，其中S是一个进程pi所有可能状态的集合。
- 系统的配置由状态向量 C={s_1,s_2,s_3,...s_n}组成。
- 一个系统的配置可以通过消息传递事件（message delivery event）或计算事件(computation event)来改变。
- 分布式系统不断经过执行C_0,ϕ_1,C_1,ϕ_2,…… 其中ϕ_i是一个计算或消息传递事件。

FSM (finite-state machine)

FSM 是表示复杂系统的数学模型 。

FSM由状态、输入和输出组成。它可以根据当前状态和接收到的输入更改状态 。

它的下一个状态由它的当前状态和它所接收的输入决定。 相同的输入可能在不同的状态下导致不同的动作。

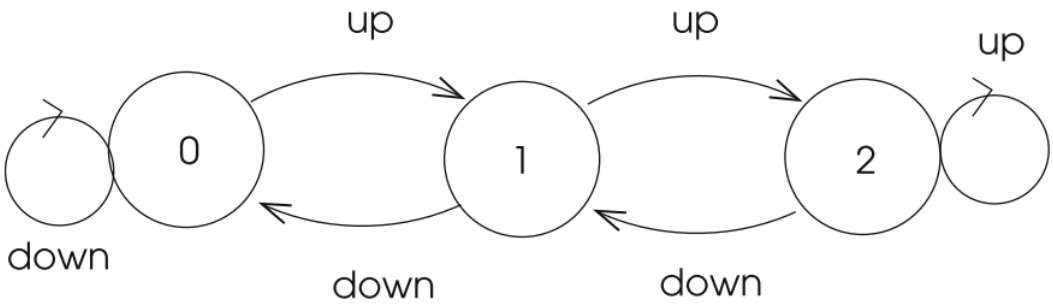
一个 deterministic FSM 是一个五元组 (I, S, S0,δ, F)

- I是一组输入信号
- S是状态的有限非空集合
- s0∈S是初始启动状态
- δ是状态转换函数,δ:S*I→S
- O∈S是输出状态的集合

FSM diagram

图中的圆圈表示其状态，状态之间的转换由有向弧表示

example: elevator



state table

FSM的状态作为行，输入作为列，表中的元素可以作为接收输入时要采取的下一个FSM状态

example: elevator

State	0(Up)	1(Down)
0	1	0
1	2	0
2	2	1

the running of the algorithm

```
#include <stdio.h>
# define UP 0
# define DOWN 1
void *fsm_tab[3][2]();
int input;
void act00(){curr_state=1;}
void act01(){curr_state=0;}
void act10(){curr_state=2;}
void act11(){curr_state=0;}
void act20(){curr_state=2;}
void act21(){curr_state=1;}
main()
{
  curr_state=0;    // initialize curr_state
  fsm_tab[0][0]=act00; // initialize FSM table
  fsm_tab[0][1]=act01;
  fsm_tab[1][0]=act10;
  fsm_tab[1][1]=act11;
  fsm_tab[2][0]=act20;
  fsm_tab[2][1]=act21;
  while (true)
  {
    printf("Type 0 for up, and 1 for down \n");
    scanf("%d", &input); *fsm_tab[curr_state]
[input];
    printf("now at floor \%d", curr_state)
  }
}
```

Performance Criteria

分布式算法的性能是根据时间、消息、空间和比特复杂性来评估的

- 时间复杂度（time Complexity）：时间复杂度是分布式算法完成顺序算法所需的步骤数。
- 消息复杂度(Message Complexity): 这个参数通常被认为是分布式算法的主要开销，因为它直接显示了网络的利用率，并指示了网络节点之间的同步开销。在网络上传输消息比在本地计算要昂贵得多。
- 比特复杂性(Bit Complexity): 消息的长度也可能影响分布式算法的性能，特别是当消息在网络中传播时长度增加时。对于一个由多个顶点和边组成的图来建模的大型网络，比特复杂度可能是非常重要的，它直接影响网络的性能 。
- 空间复杂度(Space Complexity): 这是所考虑的算法在分布式系统节点上需要的存储空间 。

Distributed Graph Algorithm Examples

Flooding Algorithm

- 发起者i向它的所有邻居发送一条消息msg(i)
- 任何接收到消息msg的节点都会将消息发送给它的邻居(除了之前接收消息的源节点)

```
1
2 { code for process i , message received from
  process j }
3 currstate ←IDLE
  //start with IDLE state
4 if i = initiator then
5     send msg(i) to N(i)
6     currstate ← VISITED
7 end if
8
```

```
 9  while true do
    //all nodes execute this part
10      receive(msg(j))
11      case currstate of
12          IDLE: send msg(i) to N(i)\ j
13                currstate ← VISITED
14          VISITED:
    //do nothing
15  end while
```

improving:

```
1  while diam(G)-- >0
2  //diam(G)is the longest path that can be traversed
   by the message msg,diameter
```

图的每条边最多将被遍历两次 ,所以消息复杂度：O(m)

时间复杂度：O(diam(G))

SpanningTree Construction Using Flooding

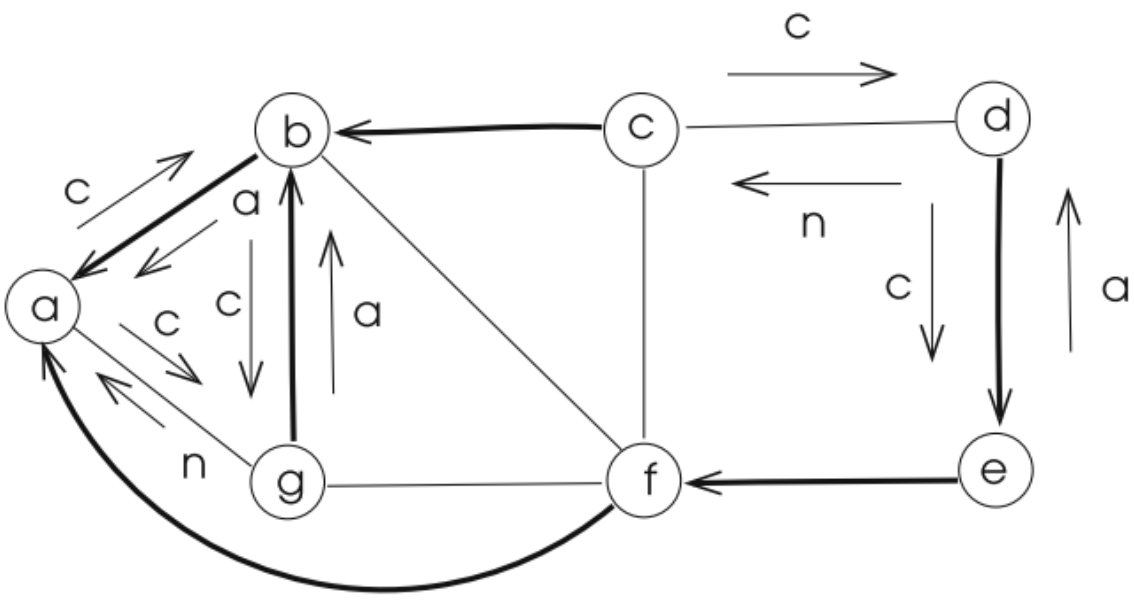
1. 有一个单一的启动程序，这个节点称为生成树的根
2. 如果第一次接收到消息 msg(j) ，则将消息 msg(j) 的发送方 j 指定为接收方 i 的父节点。
3. **ack(i):** 父节点需要知道它的子节点，所以子节点 i 应该向它的父节点 j 发送一个确认消息 ack(i) 来通知 j 这种情况
4. **nack(i):** 如果节点 i 已经有一个父节点，意味着它之前已经被访问过，它将向节点 j 发送一个否定的确认消息 nack(i)
5. **check:** 检查节点状态
6. **state:** `IDLE` `VISITED`
7. 任何节点的活动在它从所有邻居(除了它第一次收到的消息的发送者)接收到ack或nack消息时结束

```
1  int parent ←∅
2  set of int childs←{∅} , others ←{∅}
```

```
message types check, ack, nack

if i = initiator then
    send check to N(i)
    currstate ← VISITED
end if

while (childs  ∪others)!=(N(i)\{parent}) do   //all nodes execute this part
    receive(msg( j))
    case currstate ∧ msg( j ).type of
        IDLE ∧ check:   parent ← j
                        send check to N(i)\ j
                        send ack to j
                        currstate ←VISITED
        VISITED ∧ check: send nack to j
        VISITED ∧ ack:   childs ← childs ∪ {j}
//j is now a child
        VISITED ∧ nack: others ← childs ∪ {j}
 //j is not a child
end while
```



每条边至少通过两次check/ack或者check/nack遍历，所以消息复杂度为：O(m)

构造树的深度最多为n-1，所以时间复杂度为：O(n)

Basic Communications

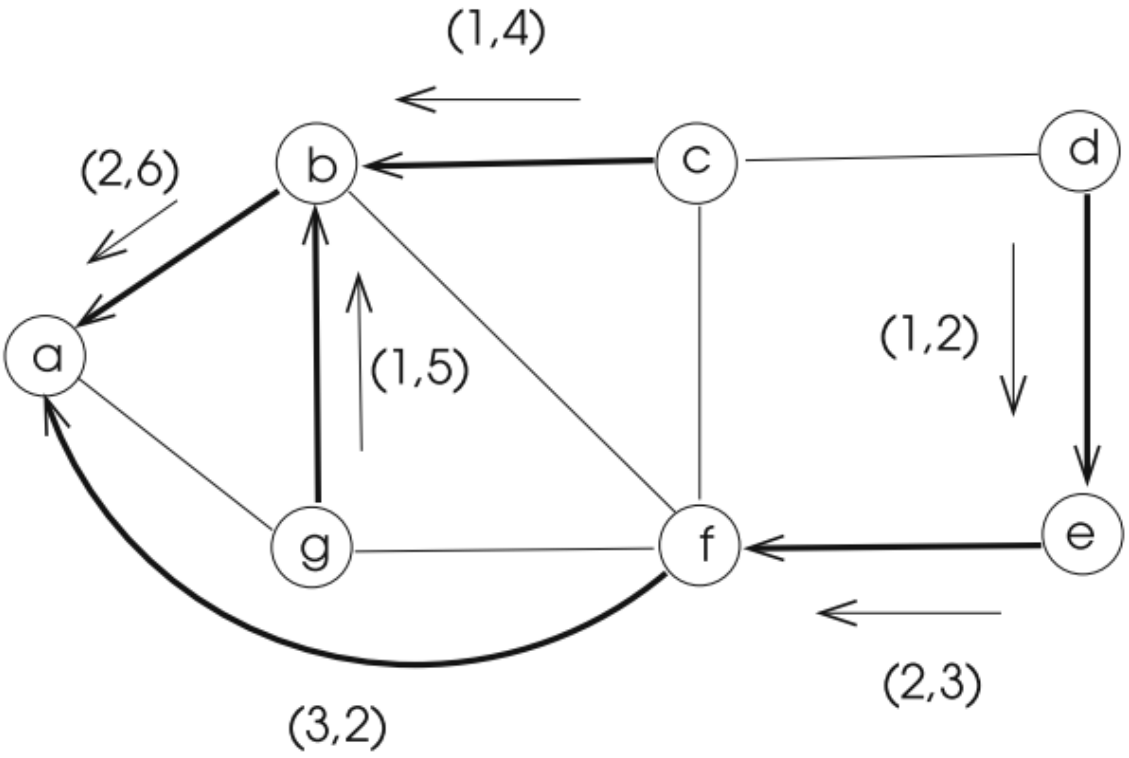
在分布式系统中执行许多基本的通信操作。 一种这样的过程是广播，该广播由节点通过发送消息来发起，并且分布式系统中的所有节点在广播操作结束时都有消息的副本。 另一个基本原语是聚合广播，其中来自每个节点的数据在系统中的特定节点处收集。

Convergecast over a Spanning Tree

1. 树的叶子将它们的数据发送给它们的父母，父母将自己的数据与叶子的数据结合起来，然后将这些数据发送给父母。
2. 中间节点实际上可以对数据执行一些简单的操作，比如求平均值或求极值。这样在树中向上发送的数据不必在每一层都变得更大。
3. 这个称为convergecast的收集过程将一直持续下去，直到在特殊节点(通常称为传感器网络中的接收器)收集所有数据。
4. 任何中间节点或根节点都等待来自所有子节点的聚合广播消息

消息用(a, b)对标记;a表示时间范围，b表示消息的持续时间

```
1: int parent
2: set of int childs, received ← {Ø} , data ← {Ø}
3: message types convcast
4:
5: if childs = {Ø} then                          ▷ leaf nodes start convergecast
6:     send convcast to parent
7: else                                          ▷ any intermediate node or root
8:     while childs ≠ received do                ▷ wait for convergecast messages from all children
9:         receive convcast(j)
10:        received ← received ∪ {j}
11:        data ← data∪ convcast(j)
12:    end while
13: end if
14: if i ≠ root then
15:     combine data into convcast
16:     send convcast to parent
17: end if
```



Leader Election in a Ring

在分布式系统中，需要领导者或协调者选举， 当算法执行中遇到故障时，领导者还可以采取补救措施。 节点和通信链路实际上可能会发生故障，尽管我们最初可以将节点指定为网络的领导者，但当发生故障时，我们仍需要选举新的领导者。 选举算法提供了在当前领导者失败时在网络中分配新领导者的途径。

考虑在具有唯一标识符的节点的环中进行领导者选举。 消息的传送仅在一个方向上。 此示例可以由具有以下状态的简单FSM方便地描述：

- LEAD：节点在此稳定状态下具有领导者。
- ELECT：当节点处于此状态时，选举仍在进行。

该算法的主要思想是 :

1. 任何检测到当前领导者故障的节点都会通过向其右侧的邻居发送包含其标识符的选举消息（假设一个顺时针单向环）来启动该算法。
2. 接收到此消息的节点将其状态更改为ELECT。 如果消息中的标识符大于其自身的标识符，则仅将选举消息传递给它的下一个

邻居。 否则，它将在接收的消息中插入大于其标识符的标识符，并将其发送给邻居。

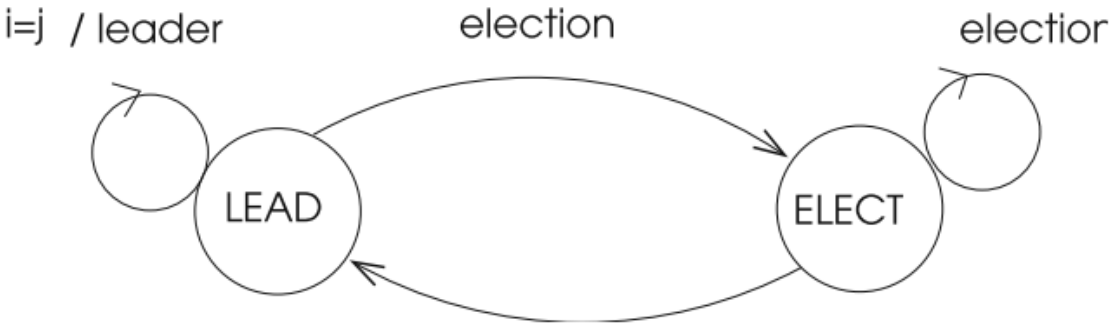
在此示例中，我们有两条消息：

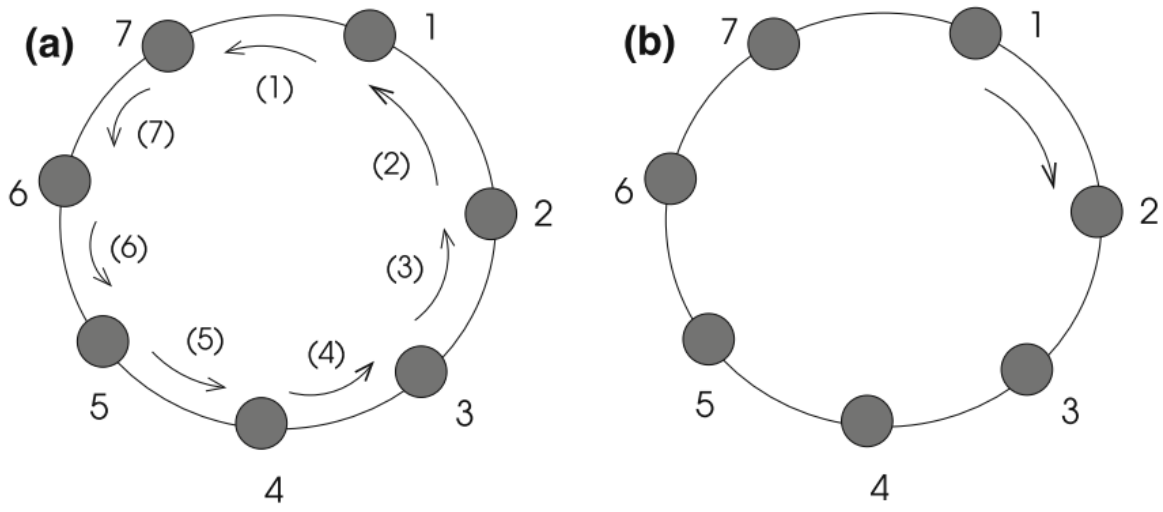
- election：由检测到领导者故障的任何节点发送。 此消息可能由多个发起者发送。
- leader：新领导者广播此消息，以通知所有节点选举已结束。

当一个具有标识符i的进程接收到一个含有标识符j的消息，会检查

- i> j：进程i在消息中用i替换j并将其传递到下一个节点。
- i <j：进程i只是将消息传递到下一个节点。
- i = j：进程i成为领导者并将领导者消息发送到其下一个邻居。

在最后一种情况下，源自节点i的选举消息已返回到自身，这意味着它在所有活动进程中具有最高的标识符。 基本上，最高标识符在所有正常运行的节点之间传输，当发起方收到自己的消息时，它确定它是领导者，并将领导者消息发送给其邻居，然后通过邻居传输将其广播到所有节点。





图a是最坏情况，最大的标识符信息遍历所有节点n次，第二大的遍历n-1次.....总共遍历n(n+1)/2

图b是最好情况，总步数为2n-1。