## Election in Arbitrary Networks

Mega-Merger

Yo-Yo

Some Considerations



## Election in Arbitrary Networks
(Gallager, Humblet, Spira '84)

### The Mega-Merger

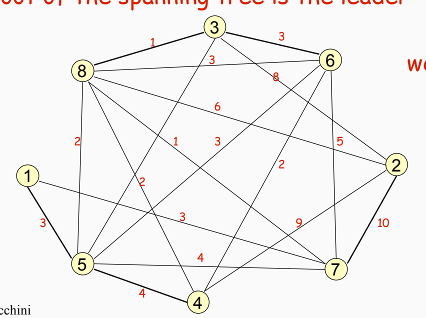
In general networks, the election problem and the spanning tree construction problem are equivalent.



## The Mega-Merger

Minimum spanning tree construction algorithm.

The root of the spanning tree is the leader



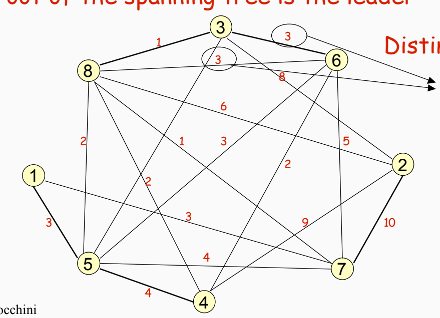
weights



## The Mega-Merger

Minimum spanning tree construction algorithm.

The root of the spanning tree is the leader

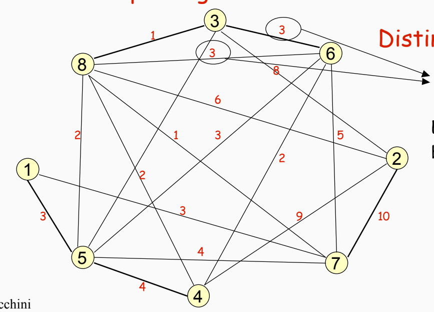


Distinct weights

## The Mega-Merger

Minimum spanning tree construction algorithm.

The root of the spanning tree is the leader



Distinct weights

(3,6)

(8,6)

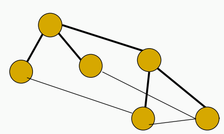
Equal weights can be made distinct



So, from now on we assume that

edges have distinct weigths





node = village with a name

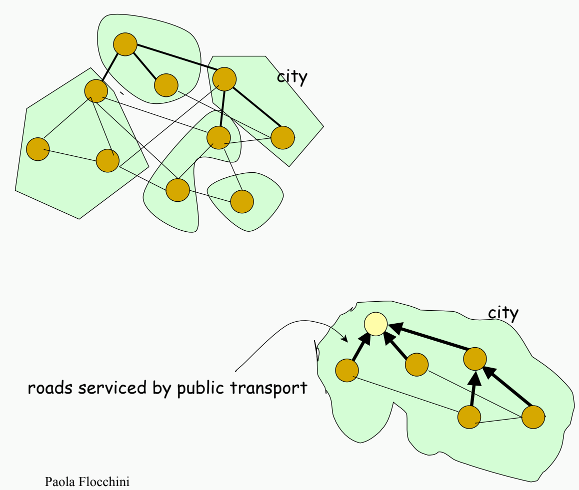
edge = road with a distance

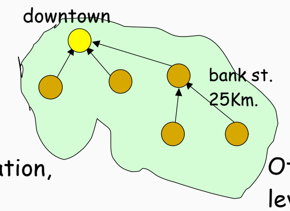
names and distances are different

**The goal**

to merge all the villages into one mega-city








1) City is a subgraph,
its spanning tree
has public transportation,
root is downtown

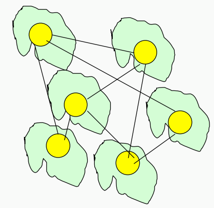
Ottawa
level=i

2) a city has a unique **name** and has a **level**
all districts eventually know the name of the city

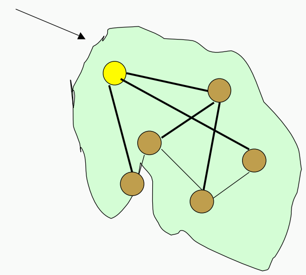
3) Edges are roads with a distinct **name** and **distance**

4) **Initially**: each node is a city with just one district and no roads. All cities are at the same level





Cities are merged into bigger cities until only ONE city is left.



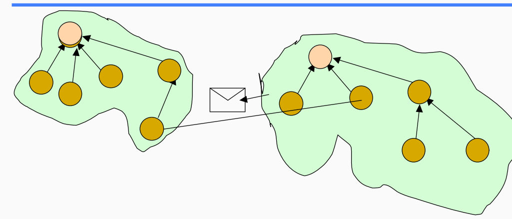


**Issues to consider when merging two cities:**

**How to name the new city**
will depend on several factors

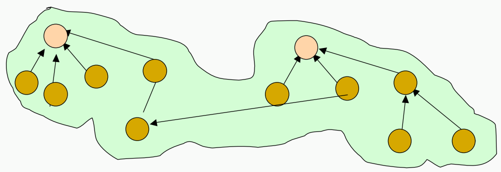
**which roads of a city will be serviced by public transports**
[the roads serviced in the two cities plus a connecting road]





5) A city must merge with the closest neighboring city.
To request a merge, it sends a *let-us-merge* message on the shortest road connecting the cities

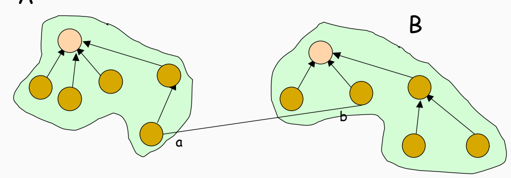
6)The decision to request a merge must come from downtown.
There cannot be more than one request at a time



7) When the merge occurs, the roads of the new city serviced by buses will be the road of the two cities + the connecting road.
The new downtown will depend on several factors.



A: city
D(A): downtown
level(A): level of city A
e(A) = (a,b): merge link with closest city (let it be B)

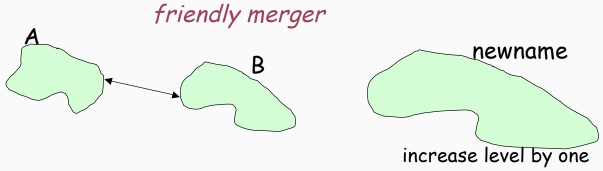



When the request arrives:
- the two cities have the same level
- the two cities have different levels

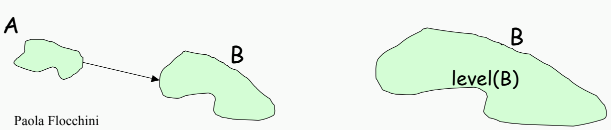


## Let A send the *let-us-merge* message to B

8) If level(A) = level(B) AND the link chosen by A is the same as the one chosen by B (e(A)=e(B)), then:
*friendly merger*



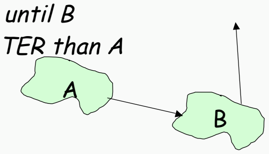

9) If level(A) < level(B) A is absorbed in B



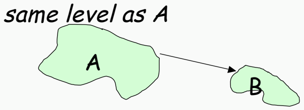


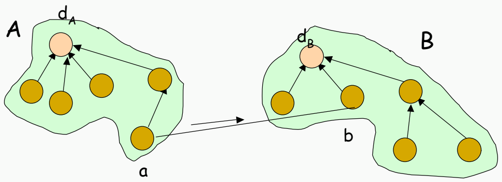

In the other cases the decision is postponed

10) If level(A) = level(B) BUT e(A) ≠ e( B), then:
*the merge is suspended until B arrives at a level GREATER than A*




11) If level(A) > level(B) then:
*the merge is suspended until B arrives at the same level as A*






**Absorption (rule 9)**

level(A) < level(B)

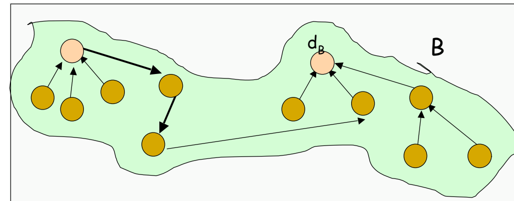
A will be absorbed by B

*b* notifies *a* about the absorption (putting B's name in the message)

a broadcast the info in A

flip all logical link direction to point to the new downtown






**Absorption (rule 9)**

level(A) < level(B)

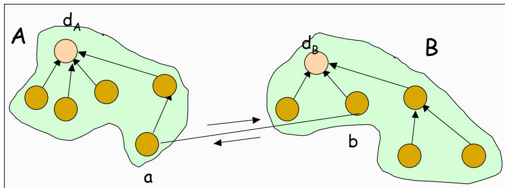
A will be absorbed by B

*b* notifies *a* about the absorption (putting B's name in the message)

a broadcast the info in A

flip all logical link direction to point to the new downtown



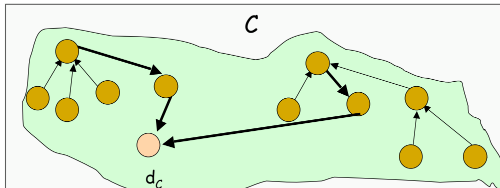



**Friendly Merger (rule 8)**

level(A) = level(B)
e(a)= e(B)

new downtown, new name, new level
downtown = min{*a*,*b*}
newlevel = oldlevel + 1
new name = name of the road connecting *a* and *b*





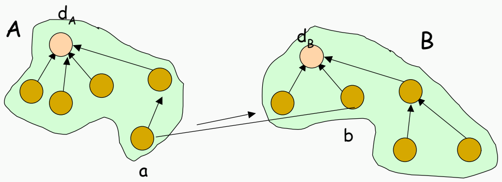

let a < b

**Friendly Merger (rule 8)**

level(A) = level(B)
e(a)= e(B)

new downtown, new name, new level
downtown = min{*a*,*b*}
newlevel = oldlevel + 1
new name = name of the road connecting *a* and *b*

*a* and *b* compute the new info independently and broadcast
the appropriate links are flipped




**Suspension (rules 10,11)**

level(A) = level(B) BUT e(A) ≠ e( B)
or
level(A) > level(B)

*b* locally keeps the necessary info for later

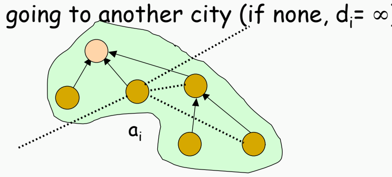
*NOTICE:* nobody in A knows about the suspension
no other request can be launched from A



## Choosing the merging link

$d_A$ needs to find the min length among all edges exiting the city

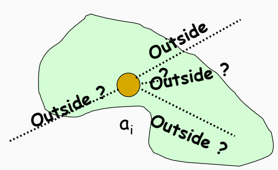
5.1) each district $a_i$ of A determines $d_i$ of the shortest road going to another city (if none, $d_i = \infty$)




5.2) $d_A$ finds the smallest
(min in a rooted tree)

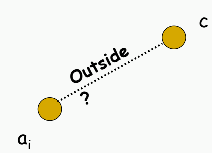


How to compute the shortest road going to another city ...




one at a time






if name(A) = name(C)
reply (internal)

if name(A) ≠ name(C)

the road is not necessarily external
(maybe C has been absorbed by A and c does not know :
in such a case level(C) < level(A))

so:

If **name(A) ≠ name(C)**
and **level(C) ≥ level(A)** then

*reply(external)*

If **name(A) ≠ name(C)**
and **level(C) < level(A)** then

*don't reply*

## More Details Discovering a friendly merger

level(A) = level(B) and e(A) = e( B)

To decide, b needs to know e(A) and e(B)

How does b know e(B) ?

e(B) is chosen by D(B), which will send the request through b

When receiving the request, b will know

So,

If e(A) =e(B), b will eventually know

If e(A) ≠ e(B), b is not the exit point, it will never know what e(B) is.



## More Details Discovering a friendly merger

level(A) = level(B) and e(A) = e( B)

**Receiving a let-us-merge:**

If b has already received a let-us-merge from D(B) to be sent to a

both b and a will know that this is a friendly merger

Otherwise

b waits

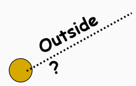
eventually, either it will know that it is a friendly merger
or its level will be increased (because of
requests from B to other cities)
and level(B) will become greater than level(A).
[absorption] [Note: A is waiting,its level cannot increase]



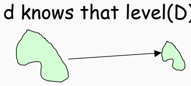
## More Details Deadlocks

waiting cases:

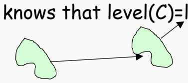
1) c send Outside? to d (level(D)< level(C)



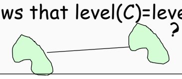

2) receiving let-us-merge on e(C)=(c,d), d knows that level(D) < level(C)



3) receiving let-us-merge on e(C)=(c,d), d knows that level(C)=level(D)
but it is not friendly



4) receiving let-us-merge on e(C)=(c,d), d knows that level(C)=level(D)
but does not know if it is friendly





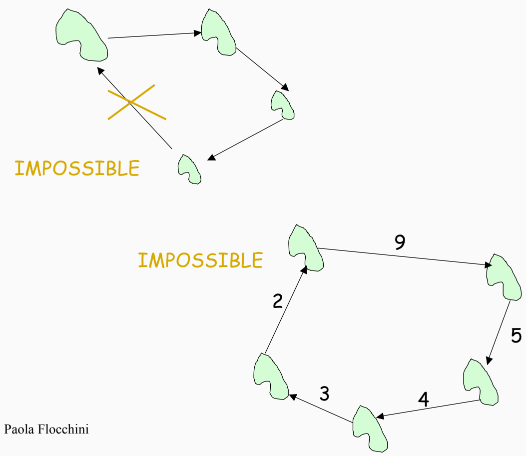

## Correctness

If a city of level l will not be suspended, its level will increase
(unless it is the mega-city)

Let city C at level l be suspended by a district d in D.
If the level of D becomes greater than l, C will no longer be suspended

No city in C will be suspended by a city of higher level

Protocol Mega-merger is deadlock-free




Paola Flocchini

## Termination

If A is the mega-city, there are no other cities.
All the unused links are internal

The minimum finding will return a special value ( $\infty$)

D(A) understands and broadcasts termination

Paola Flocchini

## Complexity

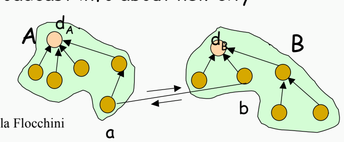
Number of messages per level: CITY A

**For each friendly merger from level i-1 to level i**

Computation of merge links: 2 (n(A)-1)

Forwarding of let-us-merge from D(A) to e(A): n(A)

Broadcast info about new city: n(A)-1




TOT: 4 n(A) - 3

Paola Flocchini

## Complexity

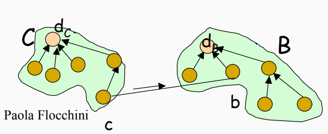
Number of messages per level : CITY C

**C absorbed at level i**

Computation of merge links: 2 (n(C)-1)

Forwarding of let-us-merge from D(C) to e(C): n(C)

Broadcast info about new city: n(C)




TOT: 4 n(C)-2

Paola Flocchini

## Complexity

disjoint cities, so: $\sum_{B \in City(i)} n(B) \le n$ $\quad City(i) = Merge(i) \cup Absorb(i)$

Number of messages per level

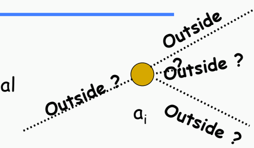
$$\sum_{A \in Merge(i)} (4n(A)-3) + \sum_{C \in Absorb(i)} 4n(C)-2 = 4\sum_{B \in City(i)} n(B) - \text{somethingsmall}$$

$$\le 4n$$

Paola Flocchini

## Complexity

Outside? with answer external




$\le n$

one at a time until found the smallest

Total Cost(i) $\le$ 5n

Paola Flocchini

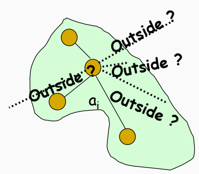
## Complexity

Useless messages Outside? with answer internal

A pair for each road that is not in the city

2(m - (n-1))

Broadcasting Termination: n-1

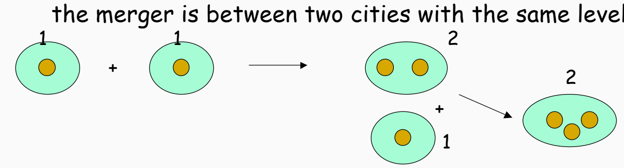



Paola Flocchini

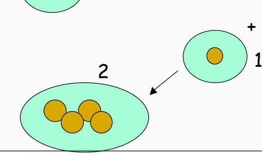
## Complexity

**How many levels ?**

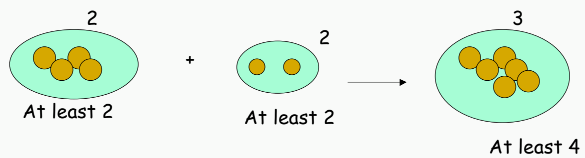
The level is incremented only if
the merger is between two cities with the same level




Level 2 there are at least 2 nodes
(maybe MORE)




Paola Flocchini




Paola Flocchini

## Complexity

In general, at Level i there are at least $2^i$ nodes (maybe MORE)

$$\text{Nodes at level } i \geq 2^i$$

$$n \geq 2^i$$

$$i \leq \log n$$

useless notification merges

**Total:** $\leq 2(m - (n-1)) + n-1 + 5n \log n$

$\leq 2m + 5\, n \log n + n+1$

Paola Flocchini