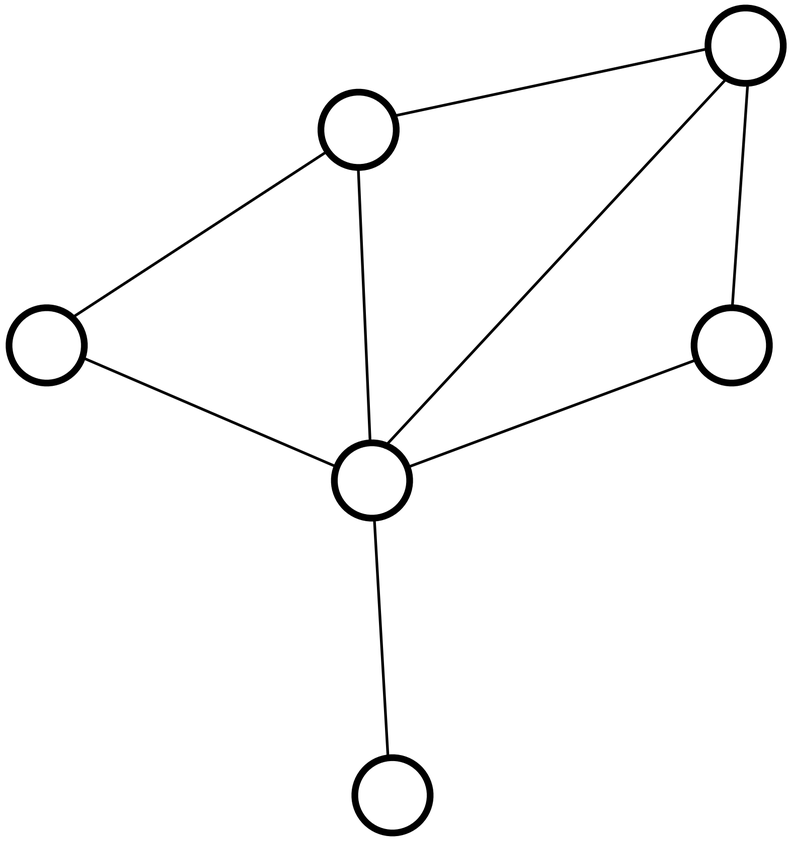
Code Competition

Path Selection Problem

Alice Burlats, Amaury Guichard, Pierre Schaus

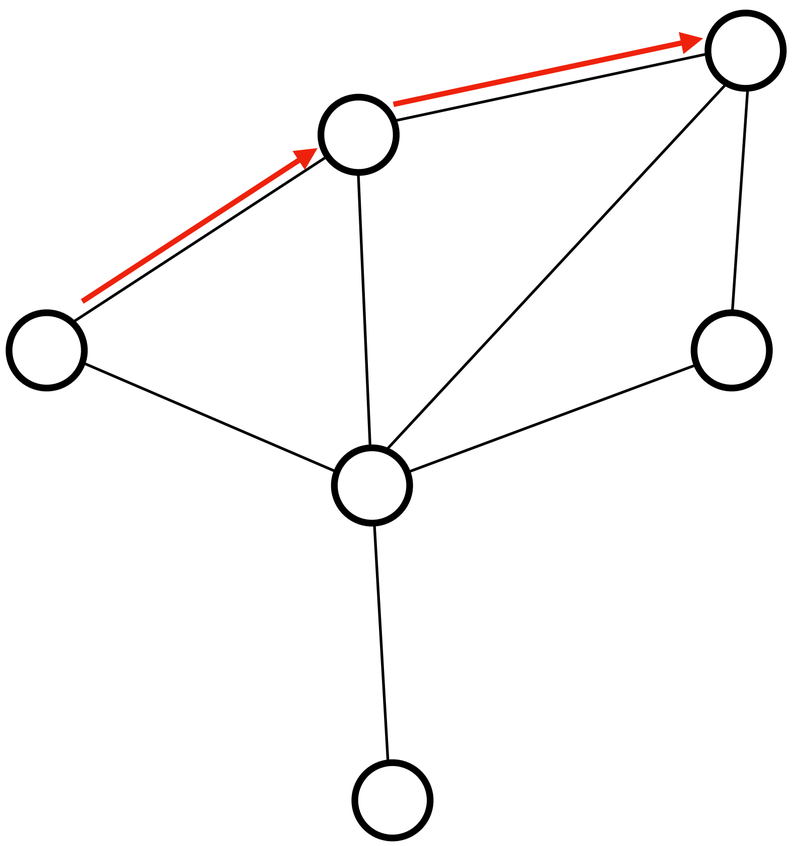
Networks and tomography

- How to know the state of each node in the network ?
- Regularly send messages on designated **measurement paths.**
- If a message doesn't reach destination → failure



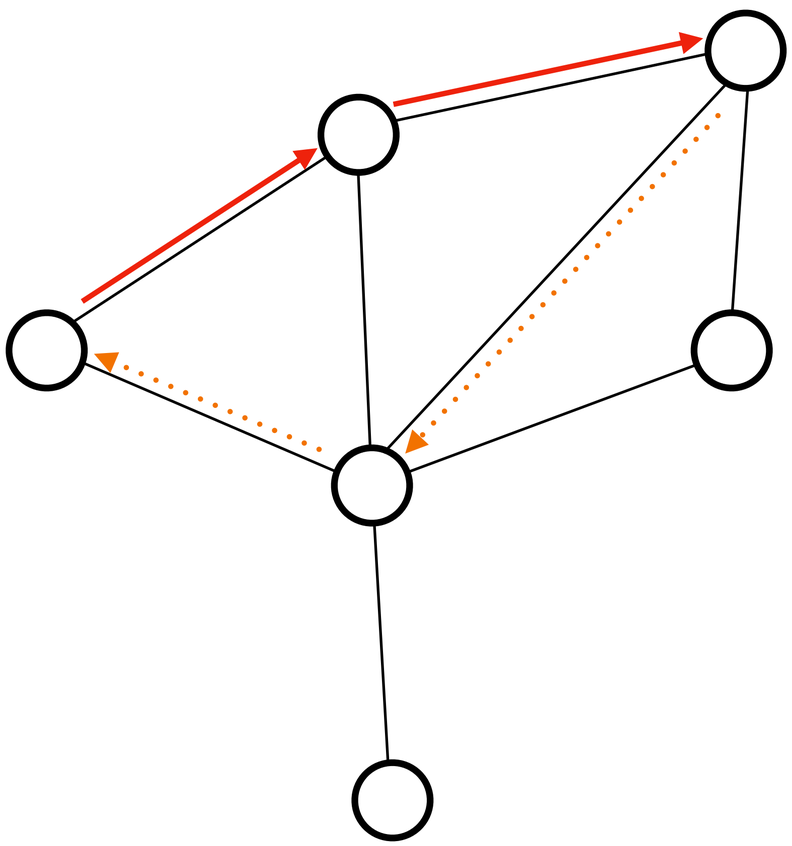
Networks and tomography

- How to know the state of each node in the network ?
- Regularly send messages on designated **measurement paths.**
- If a message doesn't reach destination → failure



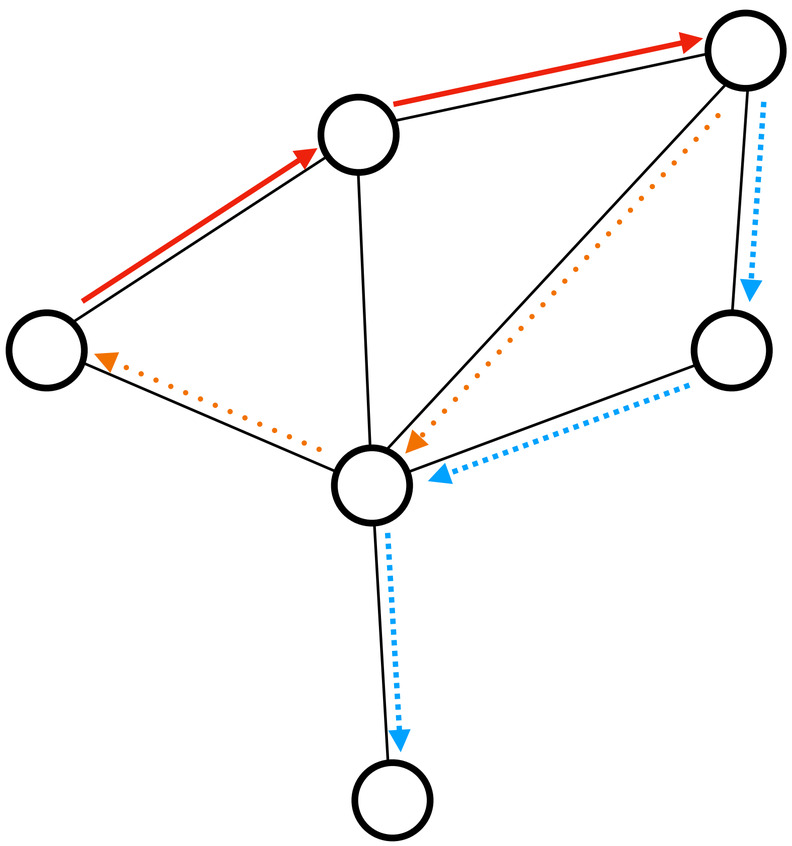
Networks and tomography

- How to know the state of each node in the network ?
- Regularly send messages on designated **measurement paths.**
- If a message doesn't reach destination → failure



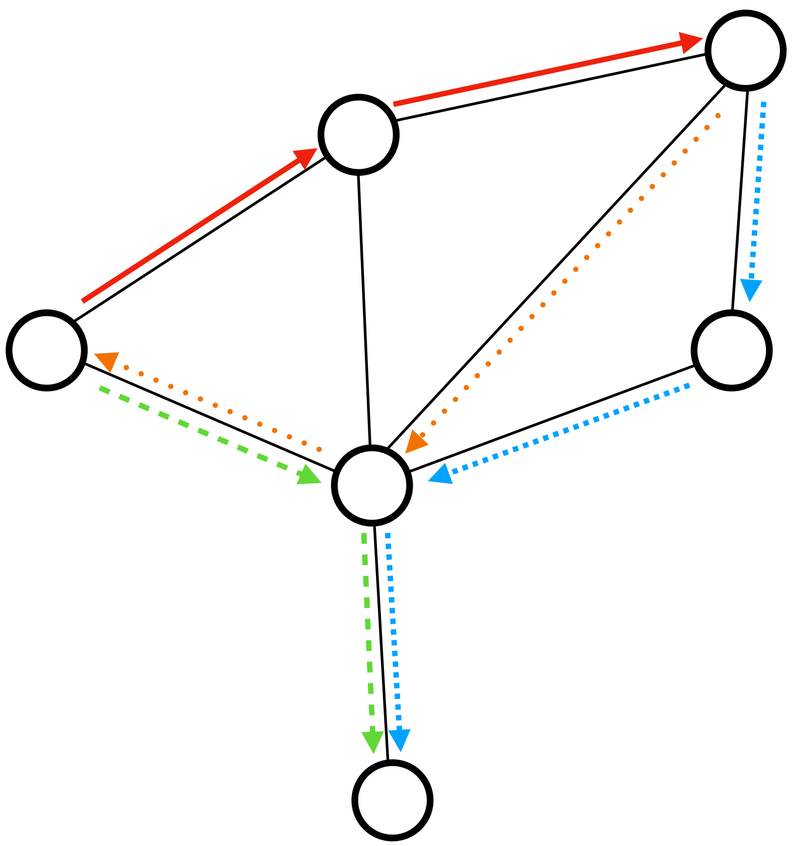
Networks and tomography

- How to know the state of each node in the network ?
- Regularly send messages on designated **measurement paths.**
- If a message doesn't reach destination → failure



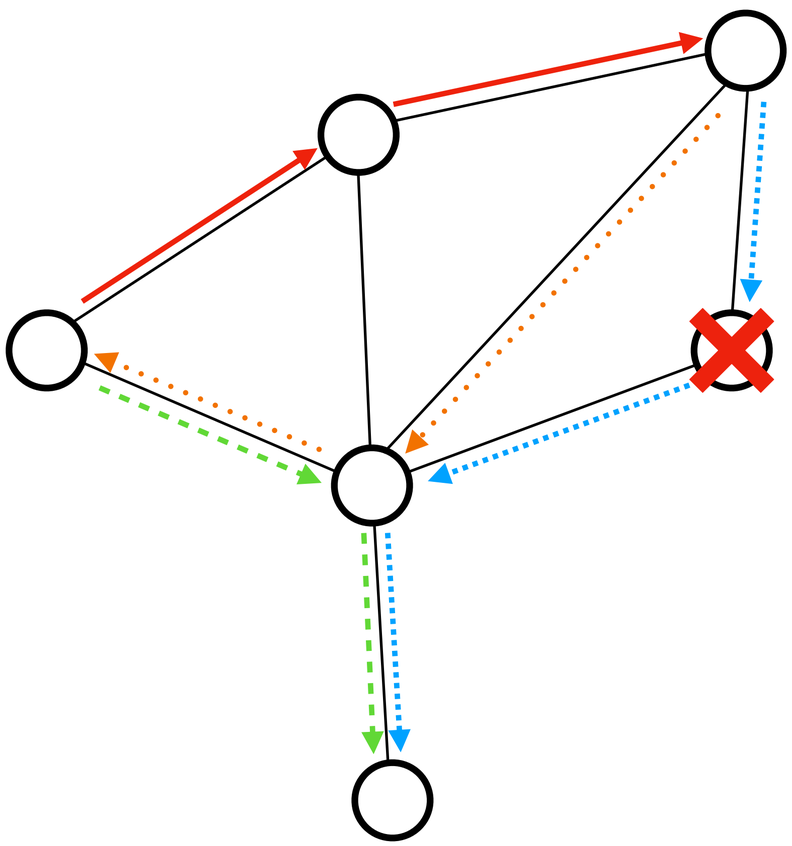
Networks and tomography

- How to know the state of each node in the network ?
- Regularly send messages on designated **measurement paths.**
- If a message doesn't reach destination → failure



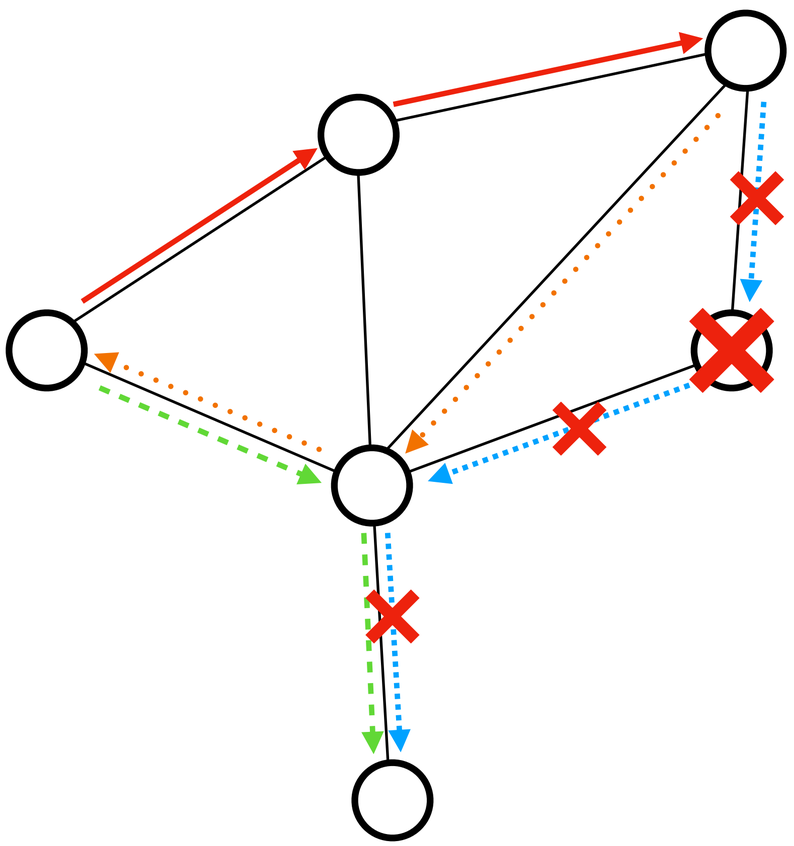
Networks and tomography

- How to know the state of each node in the network ?
- Regularly send messages on designated **measurement paths.**
- If a message doesn't reach destination → failure

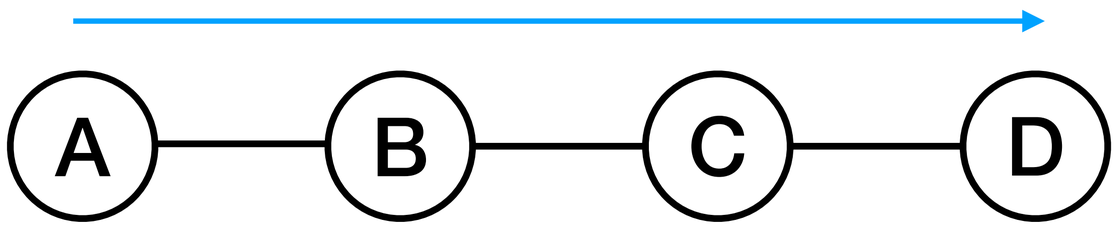


Networks and tomography

- How to know the state of each node in the network ?
- Regularly send messages on designated **measurement paths.**
- If a message doesn't reach destination → failure

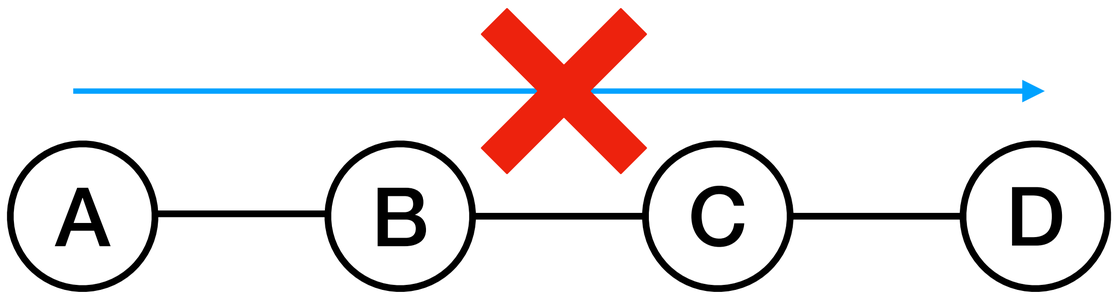


1-Identifiability



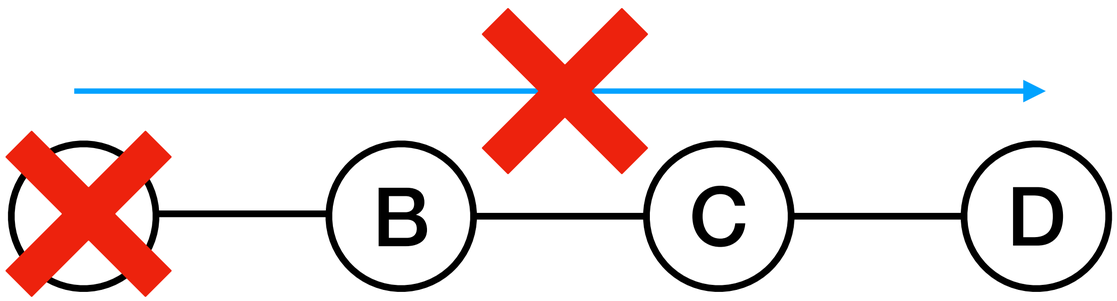
- **A** sends message to **D**, but **D** doesn’t receive it
  - -> there is a failure
- But where ?

1-Identifiability



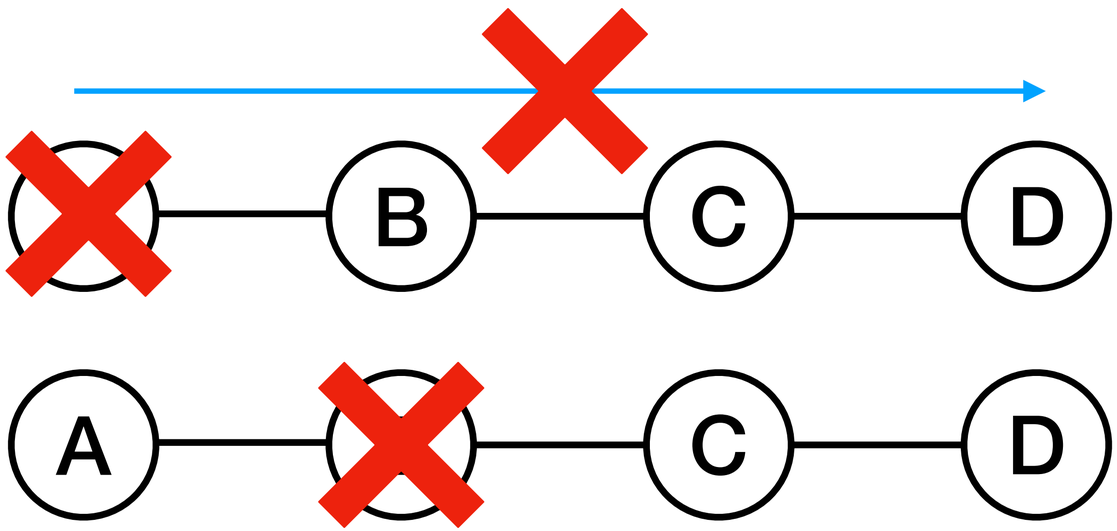
- **A** sends message to **D**, but **D** doesn't receive it
  - -> there is a failure
- But where ?

1-Identifiability



- **A** sends message to **D**, but **D** doesn't receive it
  - -> there is a failure
- But where ?

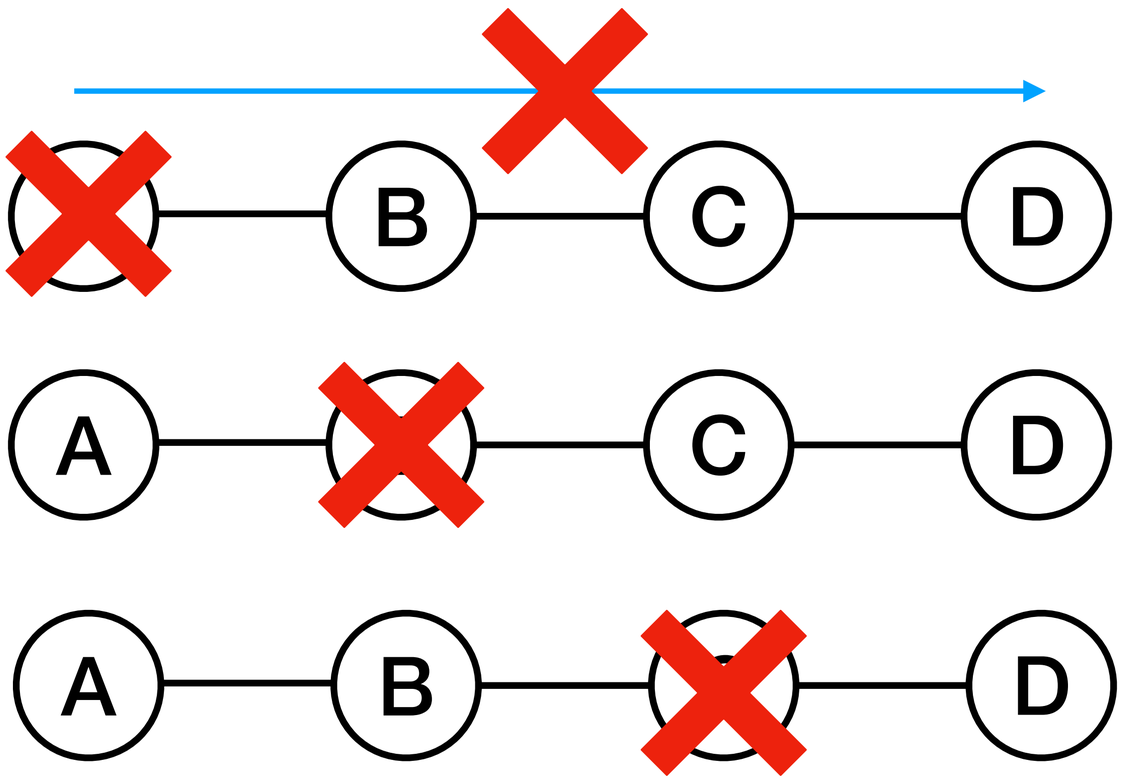
1-Identifiability



- **A** sends message to **D**, but **D** doesn't receive it
  - -> there is a failure
- But where ?

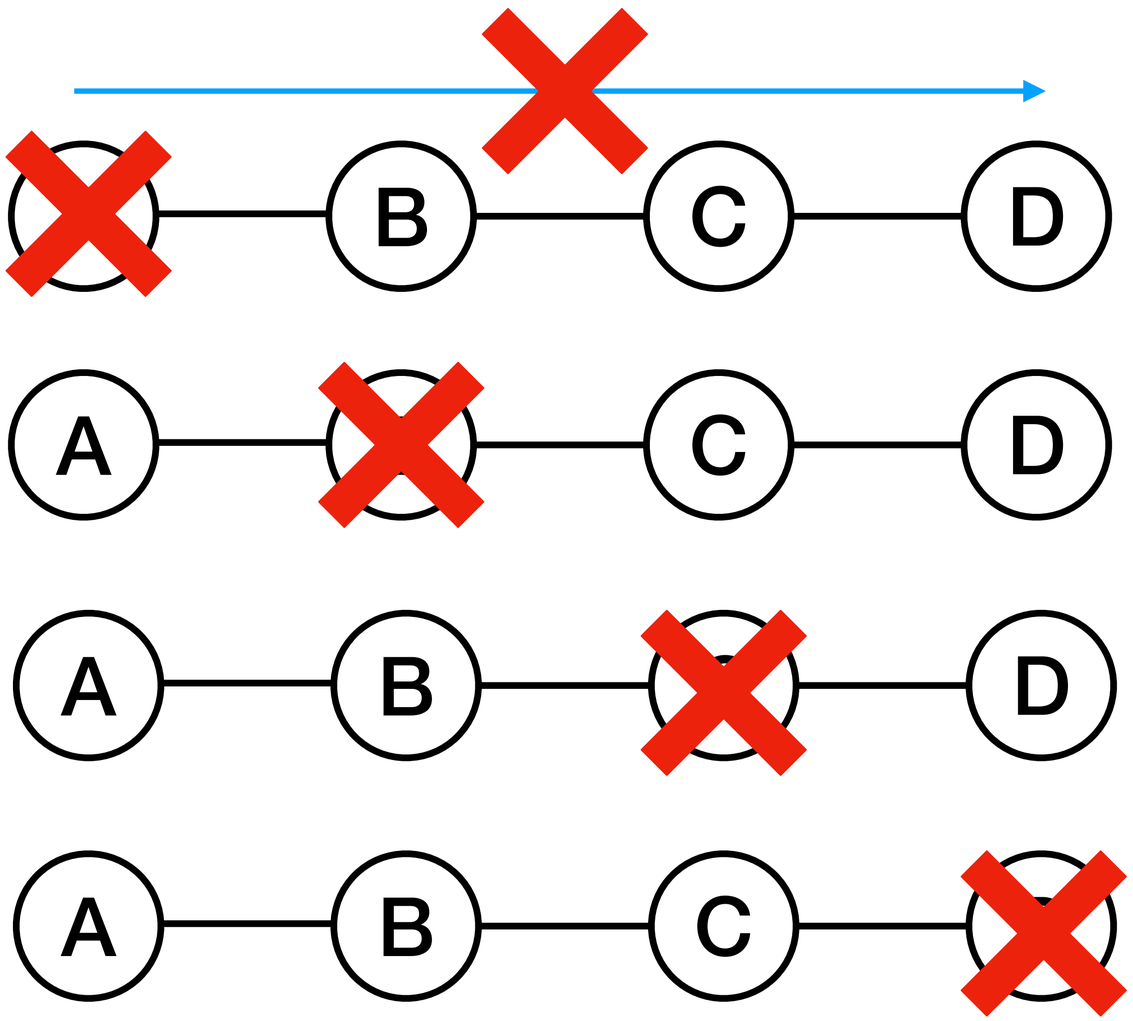
1-Identifiability

- **A** sends message to **D**, but **D** doesn't receive it
  - -> there is a failure
- But where ?



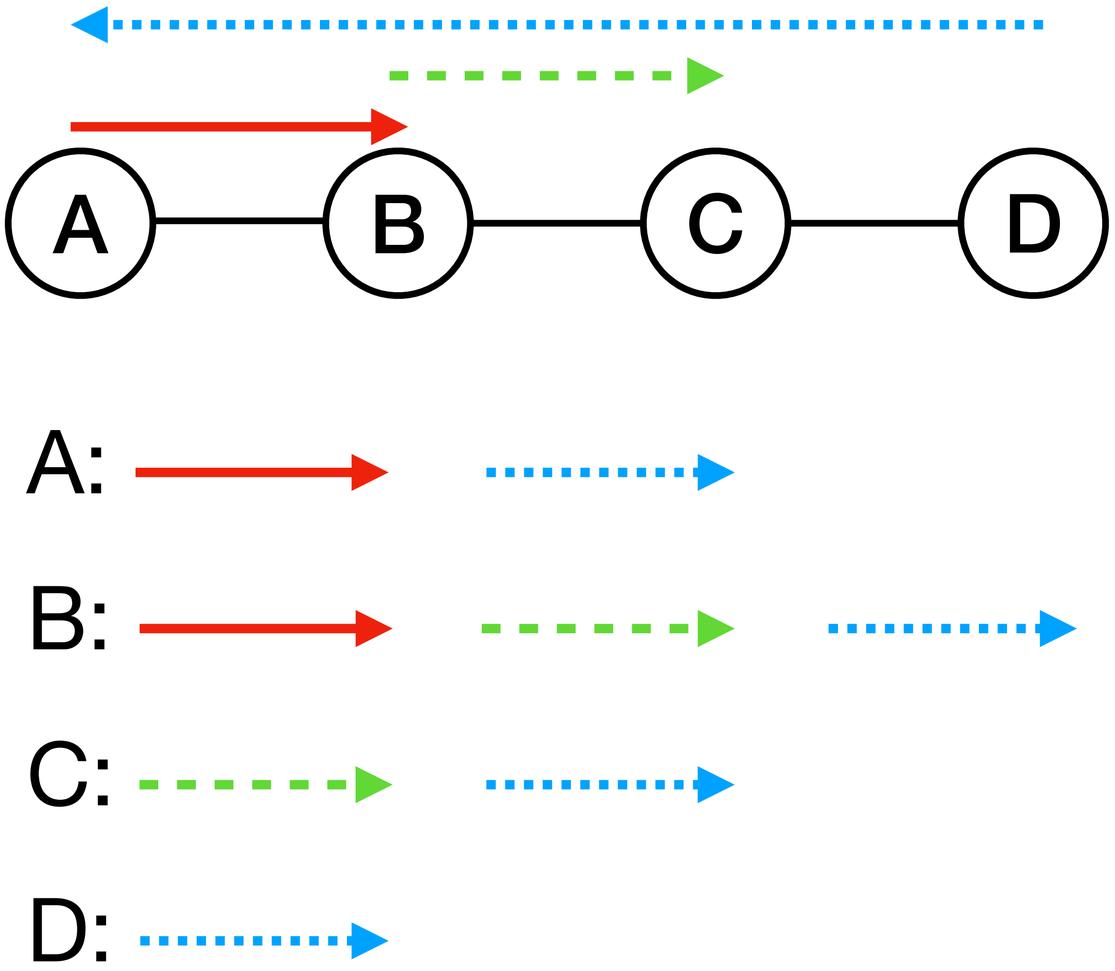
1-Identifiability

- **A** sends message to **D**, but **D** doesn't receive it
  - -> there is a failure
- But where ?



1-Identifiability

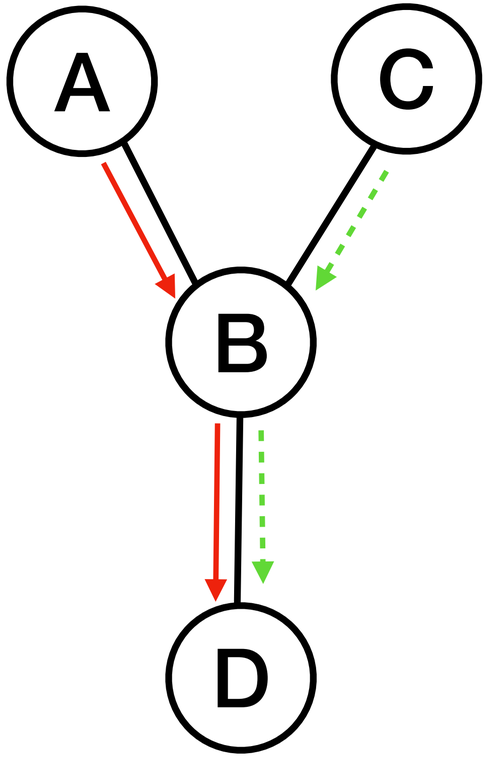
- 3 different measurement paths are used
- Now each node is crossed by a **unique** set of paths



Definition

- Coverage : Each node in the network is crossed by at least one measurement path
- 1-identifiability : Each node in the network is crossed by a **unique** set of measurement path
- 2 nodes are **distinguishable** iff their sets of measurement paths crossing them are different
- A node is 1-identifiable iff it is distinguishable from each other node

1-Identifiability



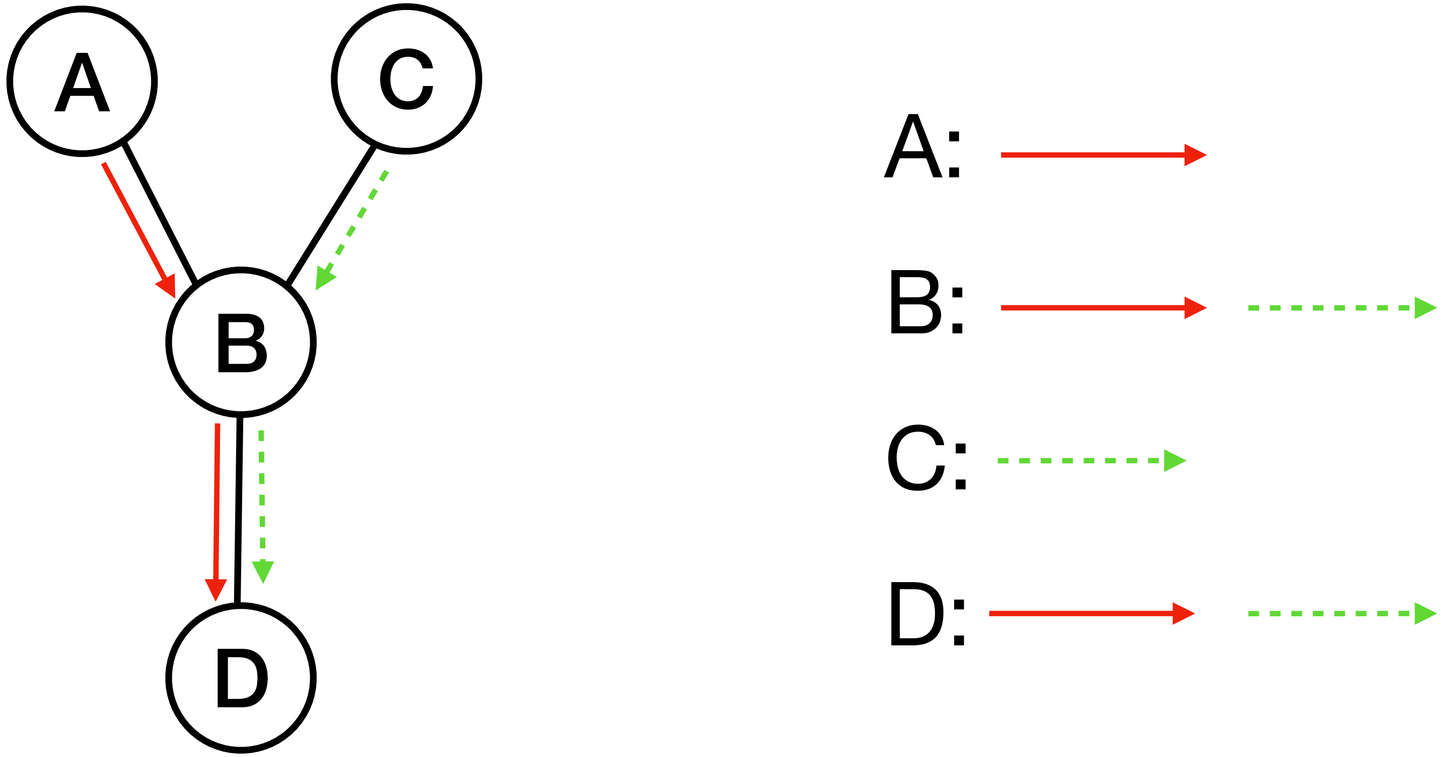
A:

B:

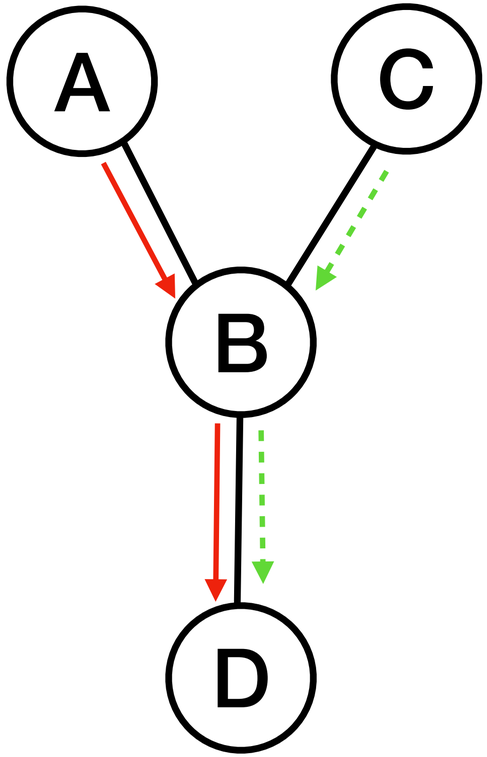
C:

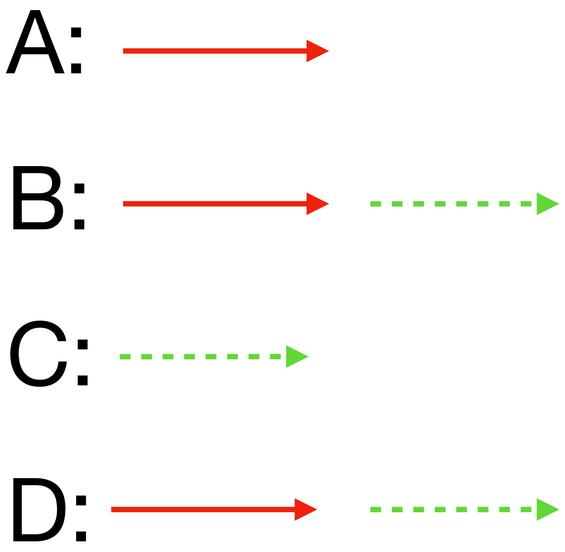
D:

1-Identifiability



1-Identifiability

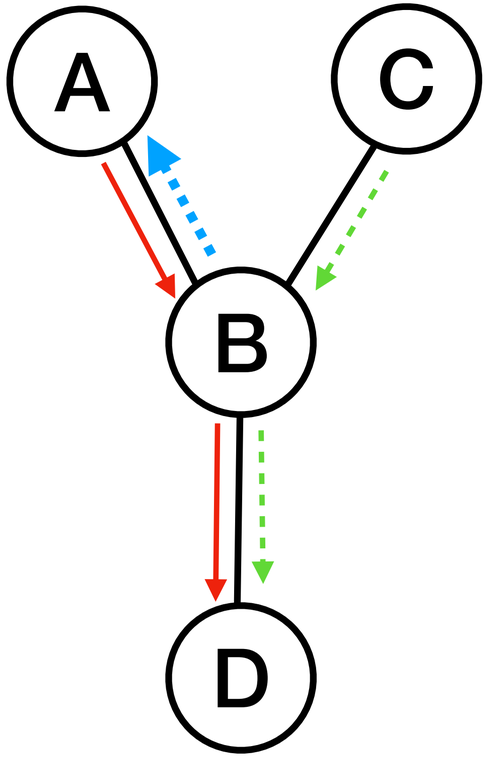


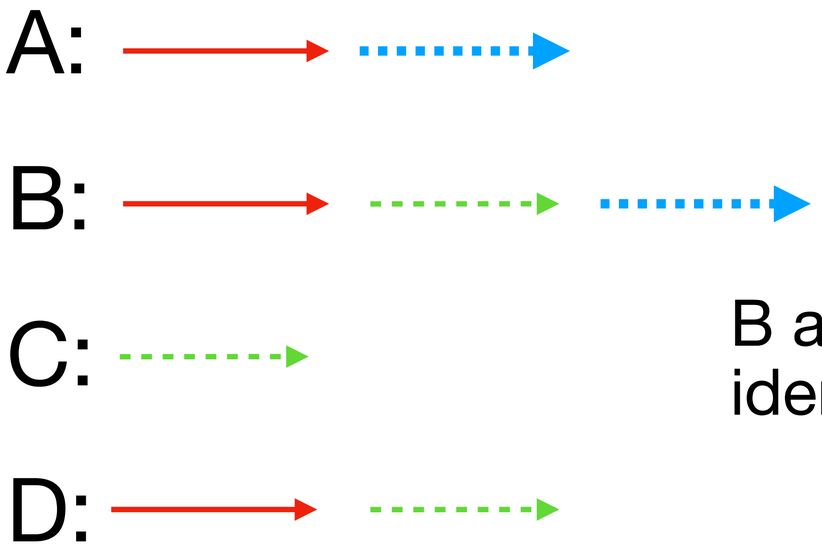


B and D are crossed by the same paths, they are not distinguishable

-> Not 1-identifiable

1-Identifiability





B and D are now 1-identifiable

Goal : **Minimize** the number of selected **measurement paths** while guaranteeing the network's **coverage** and **1-identifiability**

Some details

- You cannot design the measurements paths yourself
- A set of available paths is given to you
- You have to pick the smallest possible subset of those paths to reach coverage and 1-identifiability
- There are two paths for each pair of nodes in the network (one for each direction, not necessarily symmetrical)
- The paths are always the shortest ones

Some details

- You can find all the useful files in src/main/java/competition
- You can use the method that you want. Extend the *Solver* abstract class to implement your own solver.
- A greedy algorithm is given to you as an example. You can start from the solution it returns or remove it.

Input

- 4 instances available in data/competition
- The instance file contains :
  - The number of nodes in the network
  - The number of available paths
  - The description of the available paths
- The parsing is already done

Number of nodes

Number of available paths

Source

Destination

Crossed nodes

```
19 342
0 3 | 0 3
0 6 | 0 6
0 10 | 0 10
0 1 | 0 3 1
0 8 | 0 3 8
0 16 | 0 3 16
0 7 | 0 6 7
0 2 | 0 10 2
0 9 | 0 3 16 9
0 14 | 0 3 16 14
0 15 | 0 3 16 15
0 4 | 0 6 7 4
0 12 | 0 3 16 14 12
0 13 | 0 3 16 14 13
0 17 | 0 3 16 14 17
0 5 | 0 6 7 4 5
0 11 | 0 3 16 14 12 11
0 18 | 0 3 16 14 13 18
```

Output

Number of selected paths

```
36
0 3 | 0 3
0 6 | 0 6
0 10 | 0 10
1 3 | 1 3
1 6 | 1 6
2 10 | 2 10
2 15 | 2 15
3 0 | 0 3
3 1 | 1 3
3 8 | 3 8
3 16 | 3 16
```

- The solution must be written in a solution.tmp file
- Follow this format
- The method *writeSolution* do it in the right format and in the right file, so **use it and do not modify it**

Same path format as before

Competition rule

- 3 instances on Inginious (different from the ones given to you)
- 5 minutes to solve each instance (CPU time)
- Make sure that your algo finish and save your best solution before the timeout
- You have until the 20/12/2024 at 4pm

Grading

- 5 (bonus) points in total
- 1 pt on the Instance 1, 2pt on the Instance 2 and 2 pt on the Instance 3
- Each instance has a threshold on the cost
  - If the returned solution has a superior cost -> 0pt on the Instance
  - Otherwise it depends on how far you are from the best solution returned by another student (check the scoreboards and ignore my scores)