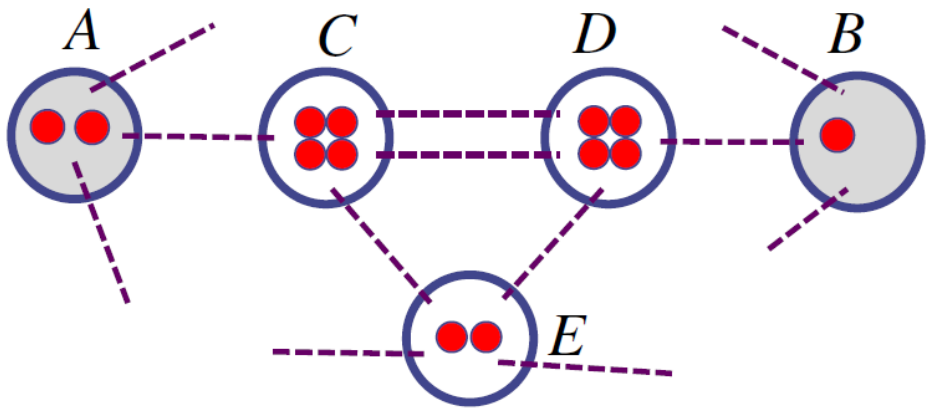
Data Structures
Programming Project #1

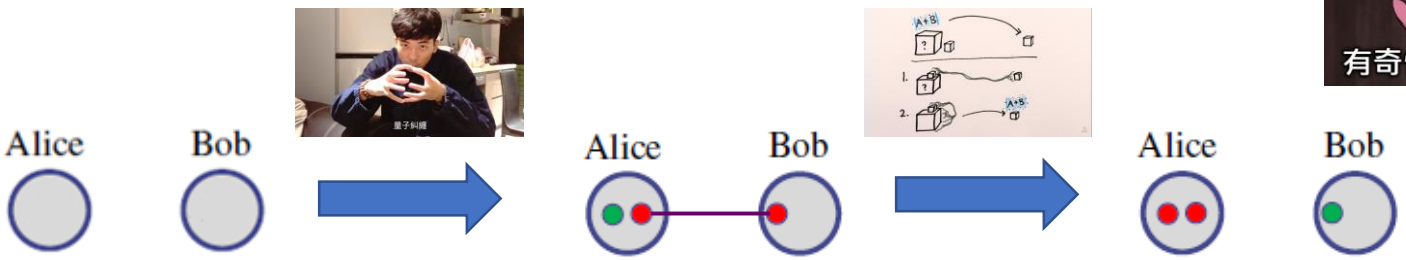
Data Transmission in Quantum Networks

- A quantum network:
- Nodes has a limited number of quantum memories
- Nodes are interconnected with a limited number of quantum channels (e.g., optical fiber)



Data Transmission in Quantum Networks

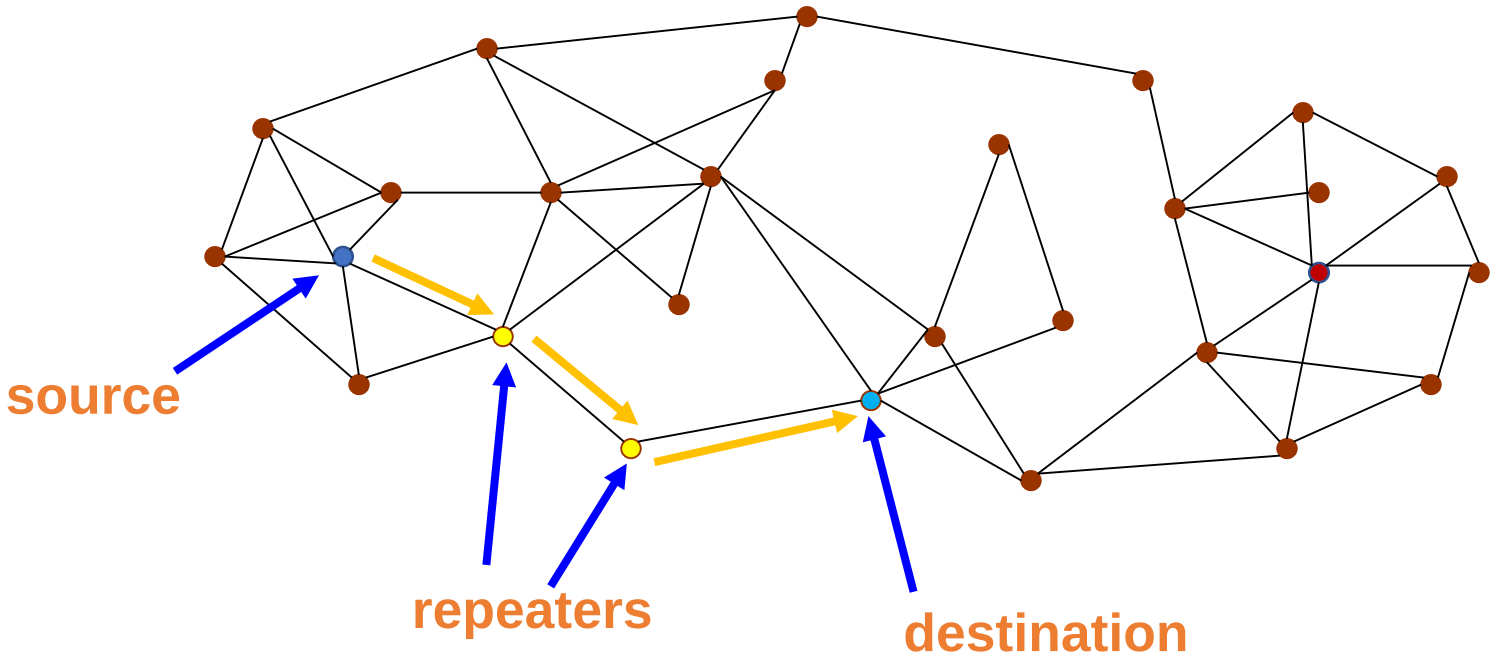
- Entangling (building an entangled link): Create an entangled pair between two nodes
- Precondition: Two nodes each with a quantum memory are interconnected with a quantum channel
- However, the success probability is $p_c = e^{-\alpha \cdot L}$, where α is a constant and L is the distance between the two nodes



https://www.popularmechanics.com/science/a25699/how-quantum-teleportation-works/

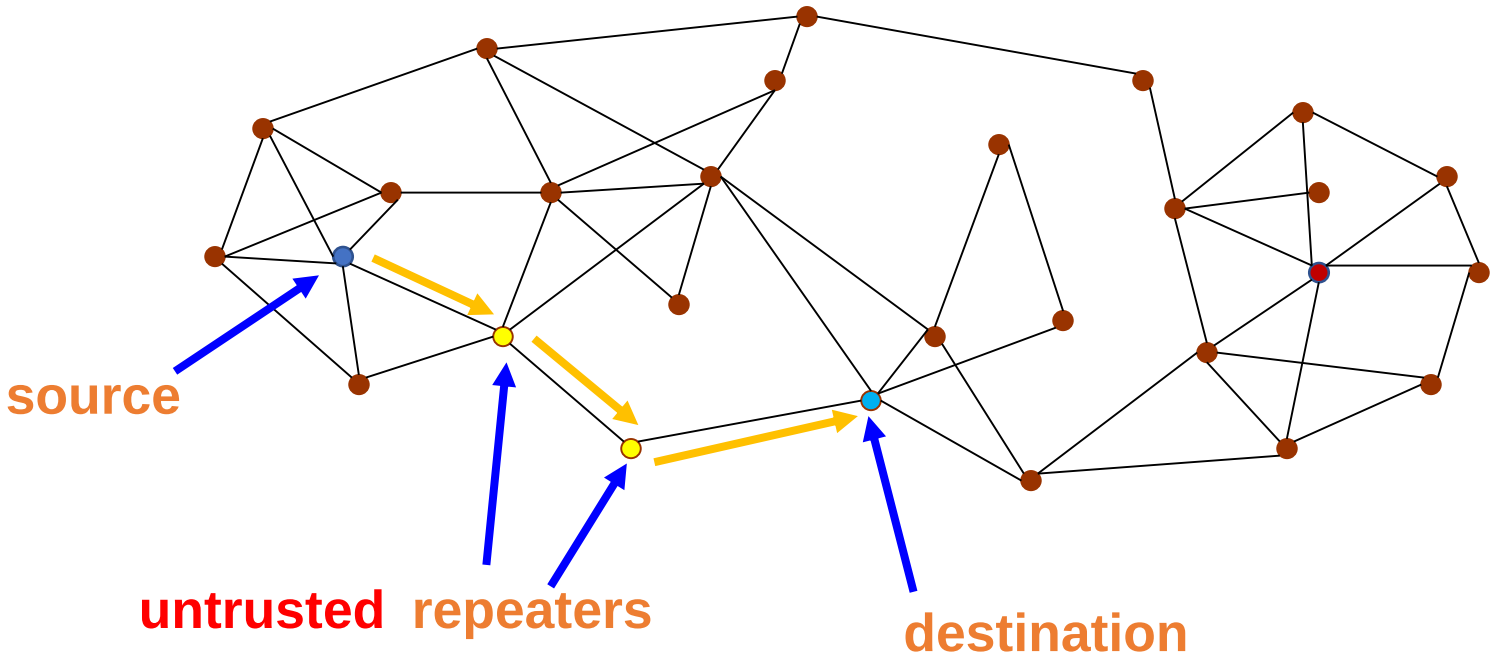
Data Transmission in Quantum Networks

- The two nodes may be distant from each other
- Classical networks:
  Repeaters use store and forward to transmit packets from a source to a destination



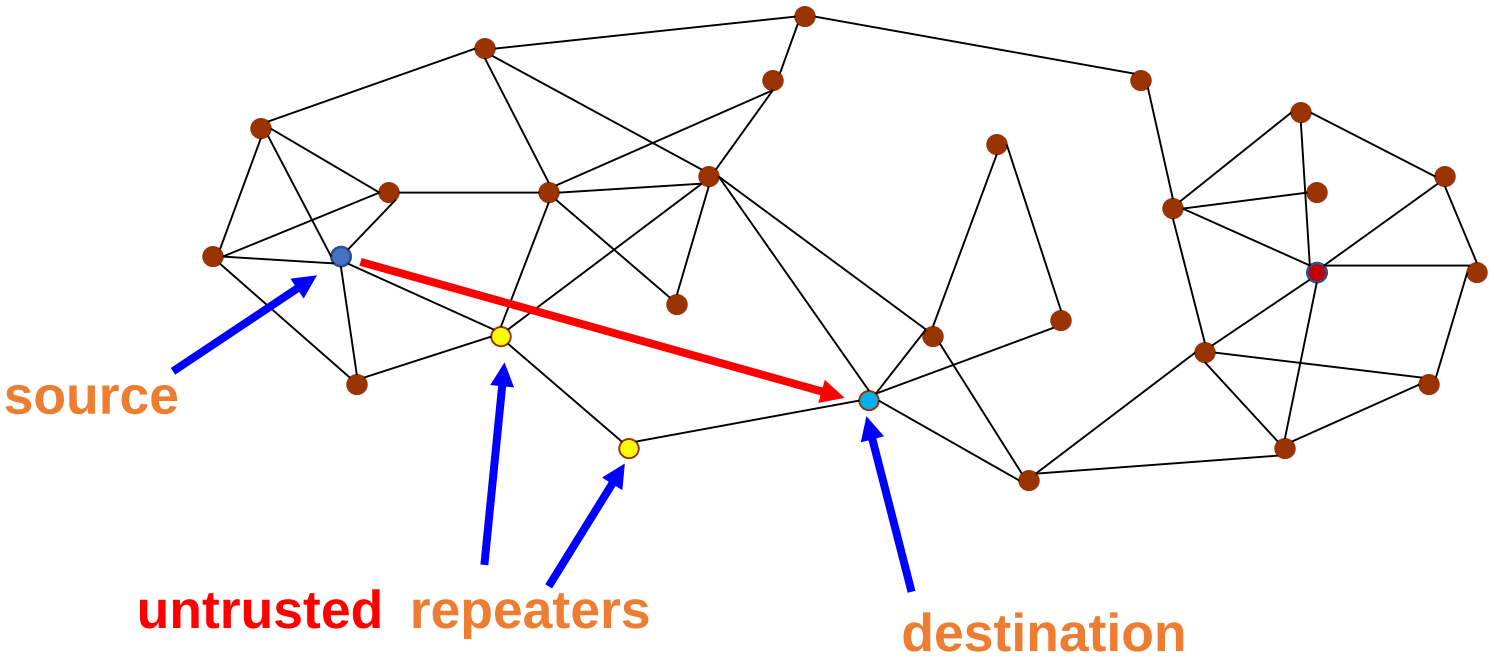
Data Transmission in Quantum Networks

- However, the data qubit may visit untrusted repeaters
- It could be destroyed, peeked at, or faked



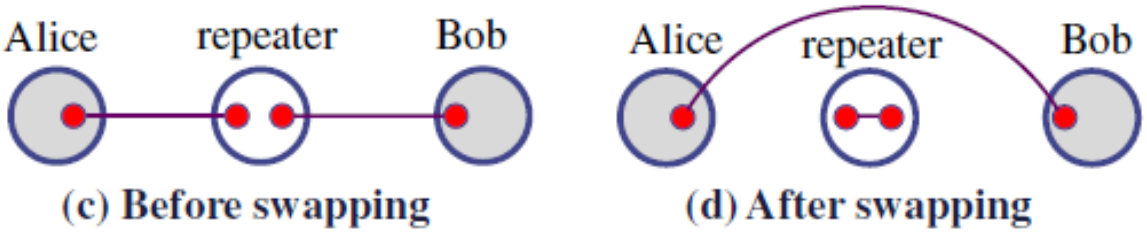
Data Transmission in Quantum Networks

- Can we send the data qubits without letting repeaters know?
- Yes, via Entanglement Swapping



Entanglement Swapping

- Alice has a data qubit for Bob
- Entangling: build the links between Alice and the repeater and between Bob and the repeater
- Swapping: build a long-distance entanglement
- However, the success probability is q

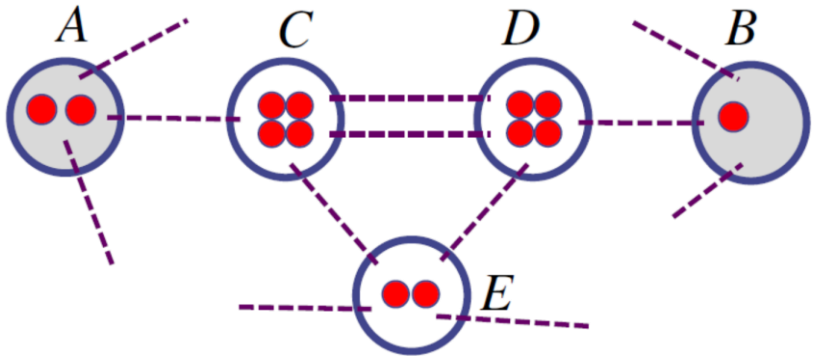


(c) Before swapping (d) After swapping

System Model & Problem Formulation

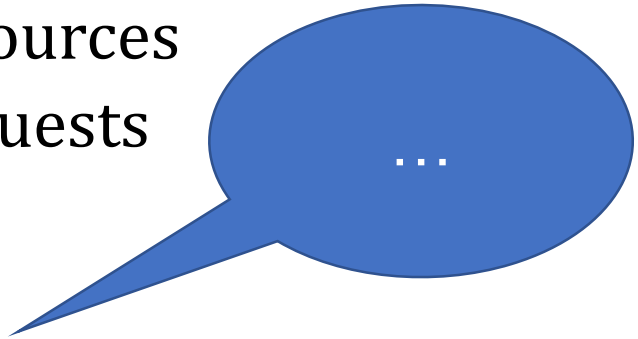
- Given:
- A quantum network with limited resources
- Multiple source-destination (SD) requests
- Constants α and β

- Goal: minimize the expected completion time
- Constraints:
- Find paths for the SD requests over time
- Limited channels
- Limited quantum memories

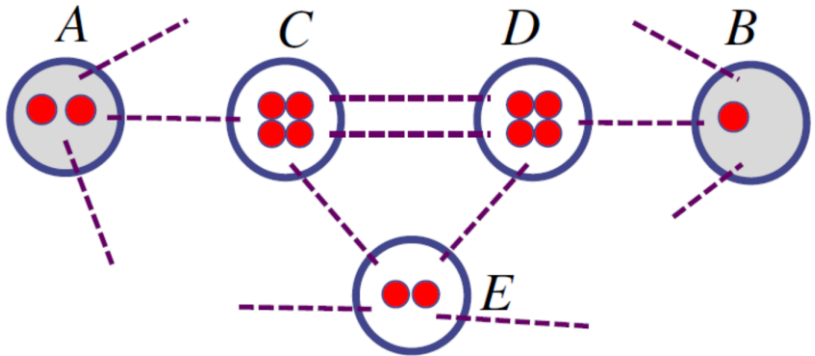


System Model & Problem Formulation

- Given:
- A quantum network with limited resources
- Multiple source-destination (SD) requests
- Constants α and β



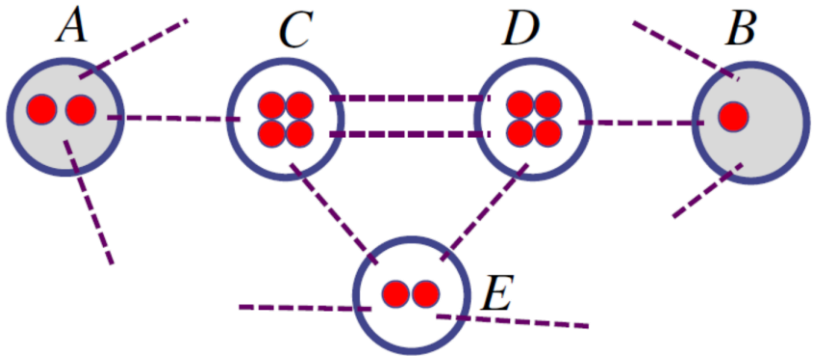
- Goal: minimize the expected completion time
- Constraints:
- Find paths for the SD requests over time
- Limited channels
- Limited quantum memories



System Model & Problem Formulation

- Given:
- A quantum network with limited resources
- Multiple source-destination (SD) requests
- Constants α and β

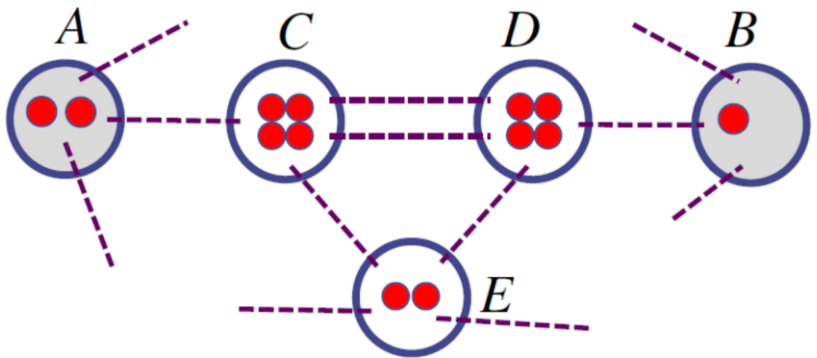
- Goal: minimize the expected completion time
- Constraints:
- Find paths for the SD requests over time
- Limited channels
- Limited quantum memories



System Model & Problem Formulation

- Given:
- A quantum network with limited resources
- Multiple source-destination (SD) requests

- Goal: maximize the number of accepted SD requests
- Constraints:
- Find a path with sufficient resource for each accepted SD request
- Limited channels
- Limited quantum memories



Bad News

- The problem is still NP-hard
- We may not always find the optimal solution in polynomial time
- Alternatively, we aim at a near-optimal solution

Programming Project #1: Entanglement Routing in Quantum Networks

- Input:
  - A node-weighted edge-weighted network $G = (V, E)$
  - Multiple SD requests
- Procedure:
  - Accept or reject each SD request
  - Find a path with sufficient resource for each accepted one
- Output:
  - The accepted SD requests and their paths
- The grade is proportional to the number of accepted SD requests

Programming Project #1: Entanglement Routing in Quantum Networks

- Input:
  - A node-weighted edge-weighted network $G = (V, E)$
  - Multiple SD requests
- Procedure:
  - Accept or reject each SD request
  - Find a path with sufficient resource for ea… …e
- Output:
  - The accepted SD requests and their paths
- The grade is proportional to the number of accepted SD requests

Programming Project #1: Entanglement Routing in Quantum Networks

- Input:
  - A node-weighted edge-weighted network $G = (V, E)$
  - Multiple SD requests
- Procedure:
  - Accept or reject each SD request
  - Find a path with sufficient resource for ea
- Output:
  - The accepted SD requests and their paths
- The grade is proportional to the number of accepted SD requests

Progra
Entang ks

- Input
  - A n (V, E)
  - Mul
- Proce
  - Acc
  - Find epted one
- Outpu
  - The
- The g
  accep

Programming Project #1:
Entanglement Routing in Quantum Networks

- Input:
  - A ne
  - Mul
- Proce
  - Acc
  - Find
- Outpu
  - The
- The g
  accep

$V, E)$

epted one

Programming Project #1: Entanglement Routing in Quantum Networks

- Input:
  - A node-weighted edge-weighted network $G = (V, E)$
  - Multiple SD requests
- Procedure:
  - Accept or reject each SD request
  - Find a path with sufficient resource for each accepted one
- Output:
  - The accepted SD requests and their paths
- The grade is proportional to the number of accepted SD requests
- We have a competition (see next page)

The Competition

- The grade is proportional to # accepted SD requests
- Basic: 75 (deadline)
  - A baseline solution (see the following pages)
- Performance ranking (decided after the deadline)
  - [0%, 50%) (bottom): +0
  - [50%, 75%): + 5
  - [75%, 90%): + 9
  - [90%, 95%): + 12
  - [95%, 100%] (top): + 15
- Homework assistant (superb deadline)
  - +10

The Competition

- The grade
- Basic: 75
  - A baseli
- Performa
  - [0%, 50
  - [50%, 75
  - [75%, 90
  - [90%, 95
  - [95%, 10
- Homewor
  - +10

SD requests

s)

deadline)

The Competition

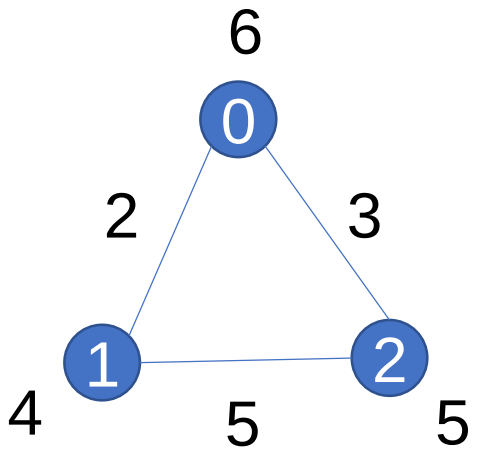
- The grade … SD requests
- Basic: 75 …
  - A baseli…
- Performa…
  - [0%, 50…
  - [50%, 75…
  - [75%, 90…
  - [90%, 95…
  - [95%, 10…
- Homewor…
  - +10

相信你們在做完作業以後

The Baseline Algorithm

- Sequentially find the shortest path for each input SD request
- If there is a tie, select the one that visits the node with a smaller ID first
  - 10 -> 4 -> 11 -> 15 -> 12 vs 10 -> 4 -> 9 -> 17 -> 12
  - Select 10 -> 4 -> 9 -> 17 -> 12
- Examine whether the shortest path has sufficient resources to accommodate the SD request
  - If yes, then accept the SD request
  - Otherwise, reject the SD request
- Repeat the above actions until all SD requests are examined

Input Sample: use scanf



Format:

#Nodes #Links #Req

NodeID #QuantumMemories

…

LinkID LinkEnd1 LinkEnd2 #Channels

…

ReqID ReqSrc ReqDst

Ex:

```
3 3 2
0 6
1 4
2 5
0 0 1 2
1 0 2 3
2 1 2 5
0 0 2
1 1 2
```

Output Sample: use printf

Format:

#AccReq
ReqID ReqSrc Rep1 Rep2… ReqDst
…

Ex:

2
0 0 2
1 1 2

Note

- Superb deadline: 9/27 Tue (adjust?)
- Deadline: 10/4 Tue (adjust?)
- Pass the test of our online judge platform
- Submit your code to E-course2
- Demonstrate your code remotely with TA
- **C Source code (i.e., only .c)**
- Show a good programming style