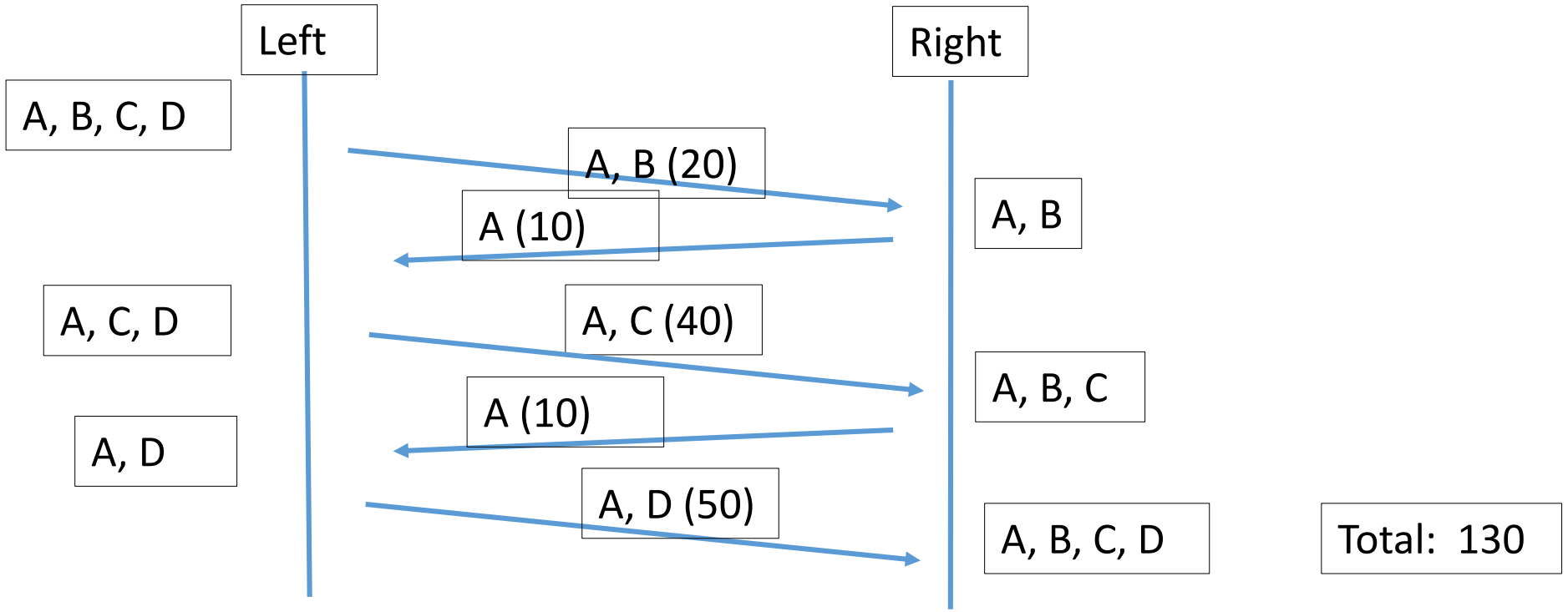
Hiking: Bridge Cross

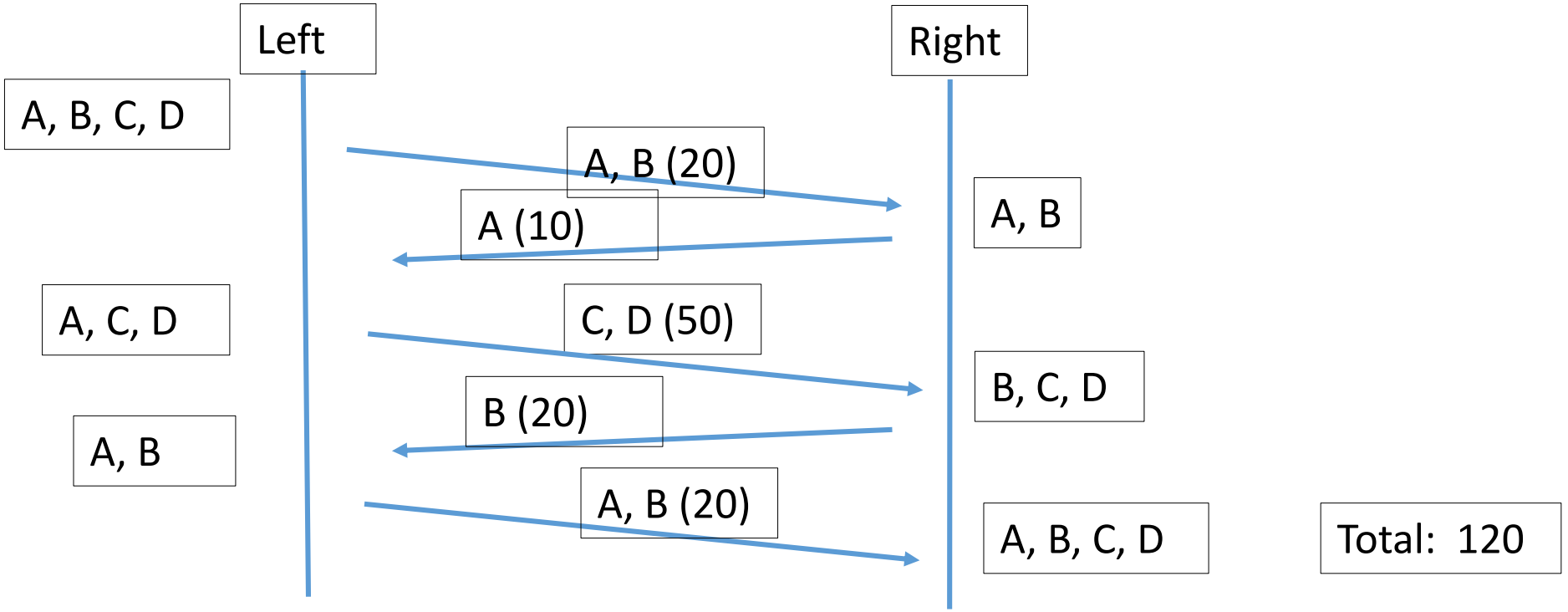
Approaches

Hikers crossing the bridge with least time can be solved with multiple approaches.
Maximum of two hikers can cross the bridge at a time. Each hiker may take different time to cross the bridge.
Time taken by two hikers to cross the bridge will be same as the slowest hiker.

e.g., Hikers: A[10], B[20], C[40], D[50]
When we look at the set, it feels like the hiker with least time to cross the bridge can be a transport companion for all remaining hikers. After crossing the bridge with other hiker he/she will return alone.
We can see this method gets the crossing done functionally, but may not give optimal time.

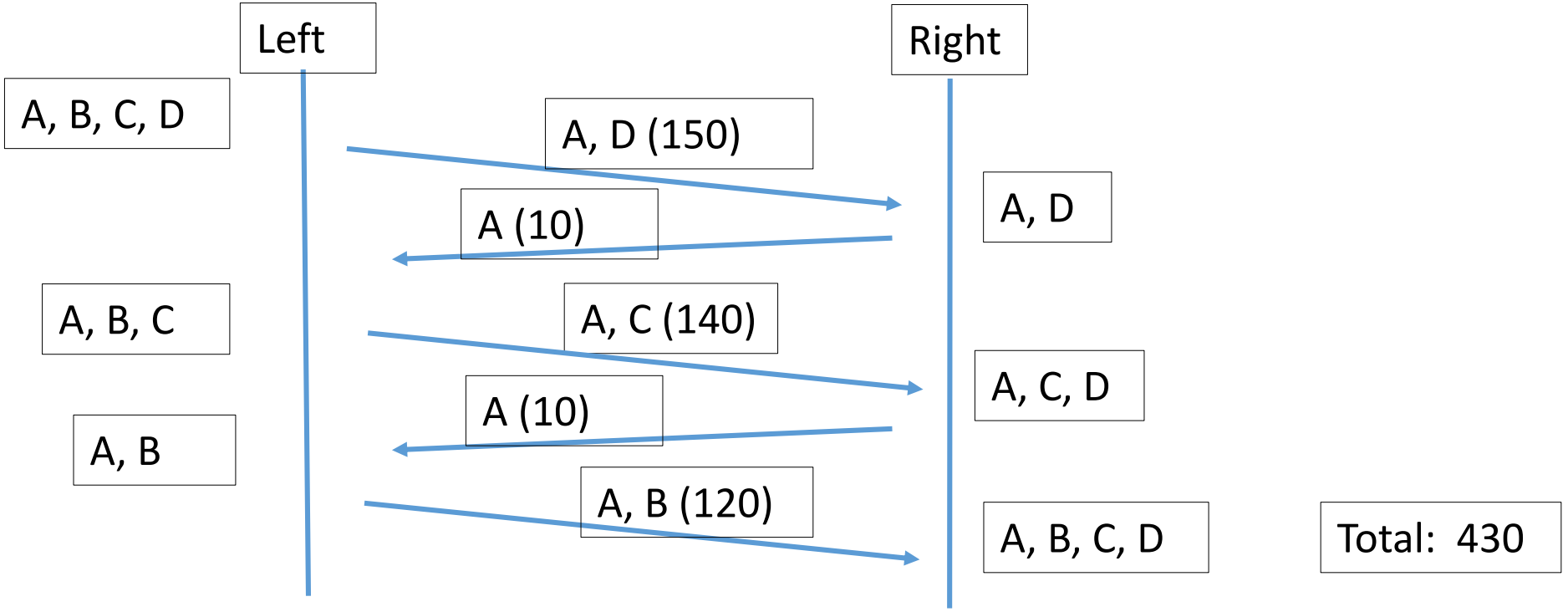


e.g., Hikers: A[10], B[20], C[40], D[50]
Previous approach took 130 t. Here is another approach where the time is optimized to 120.



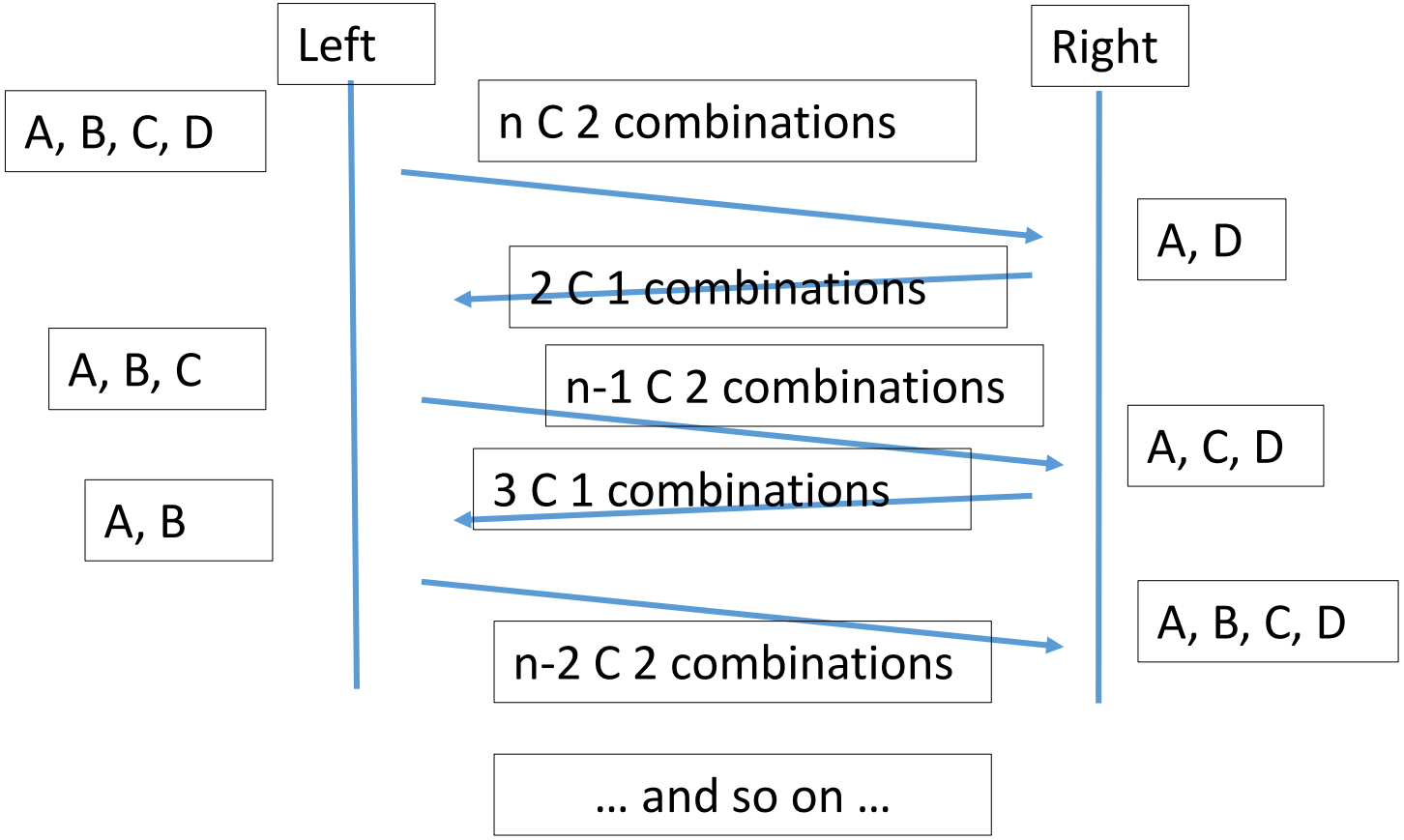
If the hikers are arranged in sorted order, we can have recursive approach to solve this. We can see a pattern of 1st and 2nd hikers with least time to cross acts as a travel companion for the remaining folks.

If the hikers time is skewed heavily where one of the hiker's time is a lot less than others, then the previous approach will not provide the optimal time (it will be 520 for the below example).
e.g., Hikers: A[10], B[120], C[140], D[150]
Here is another approach where the time is optimized to 430.

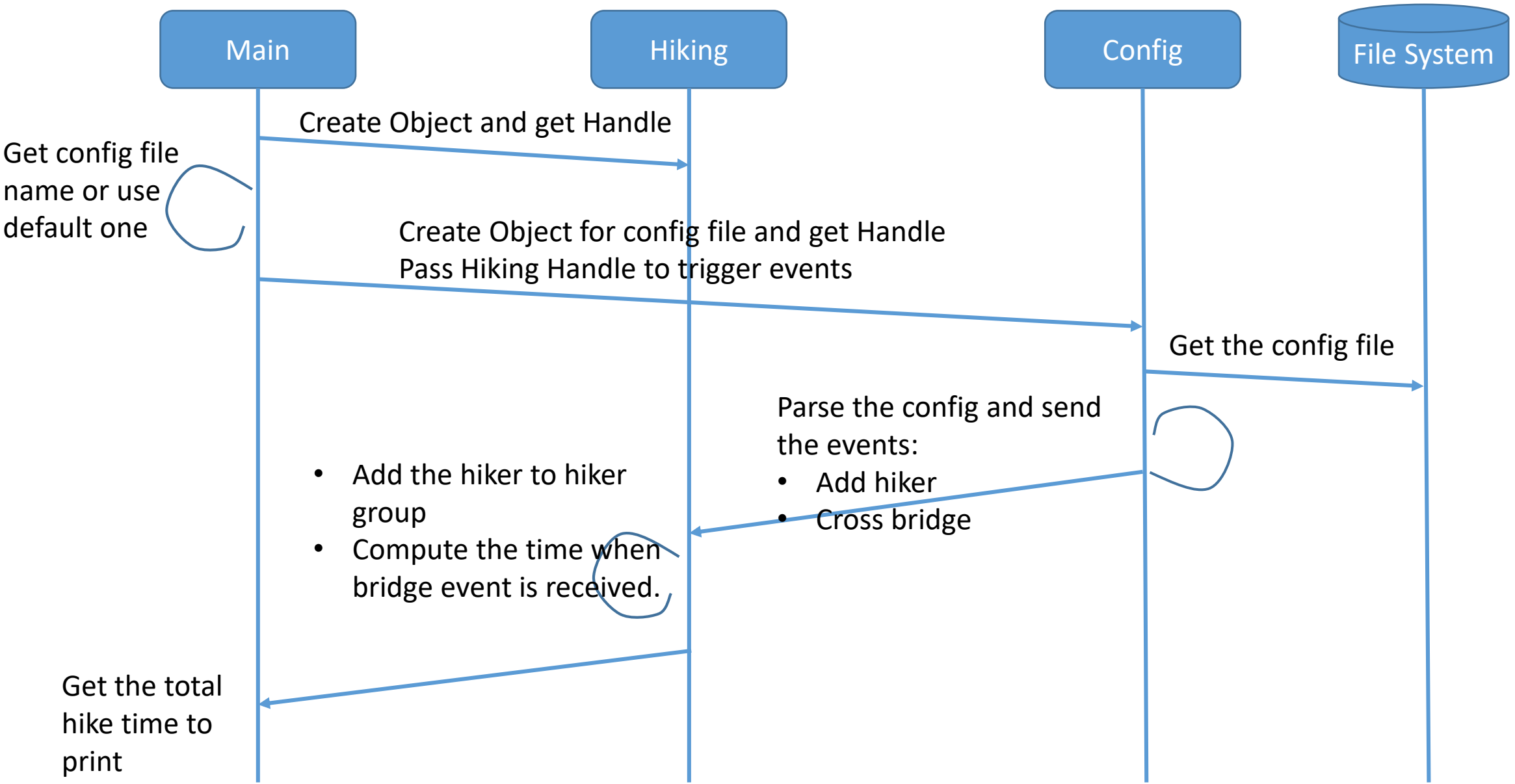


If the hikers are arranged in sorted order, we can have recursive approach to solve this. We can see a pattern of 1st hiker with least time to cross acts as a travel companion for the remaining folks and picks remaining folks from the end of the list.

We can also use the approach to enumerate all combinations of the hiker pairs to cross the bridge and keep track of the combination which provide the least amount of time compared to all other enumerations.
This is highly exhaustive as all combinations of two hikers from the left bridge and all combination of one hiker from the right bridge will be used to enumerate. Time complexity is O(n! * n-1! * n-2! ….) as shown below



Design


Main
Hiking
Config
File System
Create Object and get Handle
Get config file name or use default one
Create Object for config file and get Handle
Pass Hiking Handle to trigger events
Get the config file
Parse the config and send the events:
• Add hiker
• Cross bridge
• Add the hiker to hiker group
• Compute the time when bridge event is received.
Get the total hike time to print

Source Code

As it is a small project, to keep it simple, all files are in one folder.
It can be reorganized to keep source, header, binary, config, obj, lib, 3rd-party, make, etc., in different directories.

In linux, use Makefile
In Windows, use the project BridgeCross.sln

Config file:

- bridge_cross_1.yaml and bridge_cross_2.yaml are used to check the sanity of two methods used
- **hiking_event_default.yaml** is the default config used.
- Default config can be changed in main.c or config.h or config file can be given as argument to main.

Config Parser:

- config.[cpp,h] implements yaml parser for this project.

Core logic:

- hiking.[cpp,h] implements the methods to compute the time required to cross bridges.

Main:

- main.cpp – entry to the project, initializes the objects, comparison of outputs of two different approaches. For detailed debug, different trace levels can be enabled here.

Helper and Debug:

- hiker.[cpp,h] & bridge.[cpp, h] – Implements helper structures.
- debug.[cpp, h] – Implements different trace levels for debugging