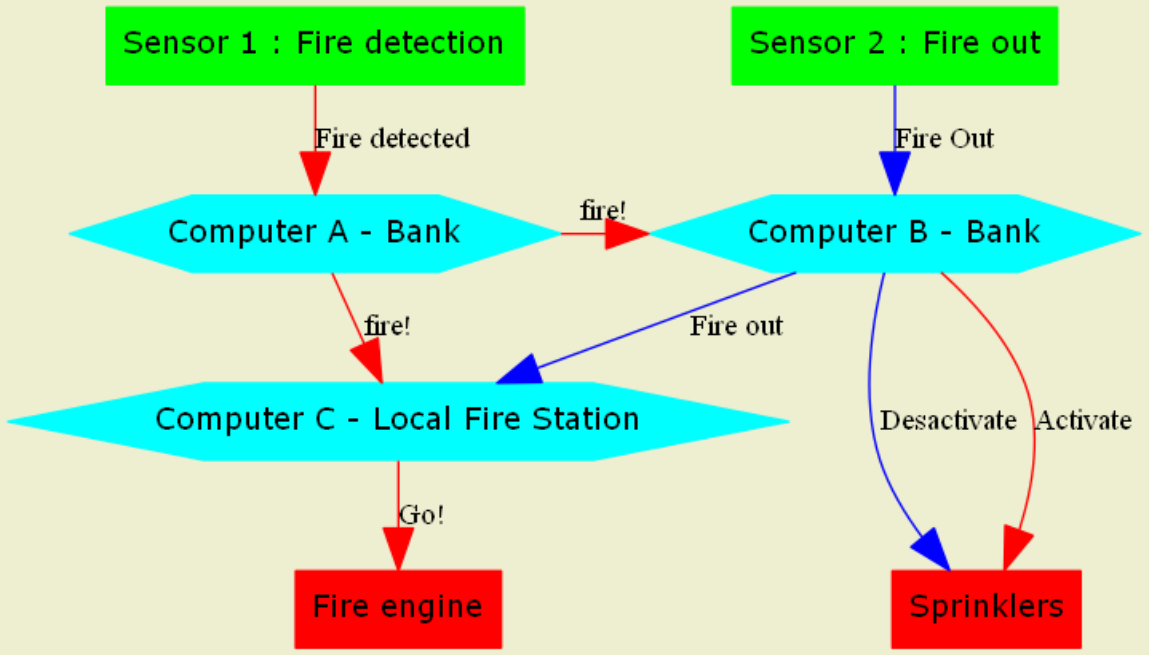
Coursework 2

Newcastle University

December 20, 2011

i) Argue on the system design

Here is a vue of the actual system :



Waiting for (2d+e) time units after a "fire!" message have been received looks like a good idea, as it correspond to :

- d : Max time for A to send message to B
- d : Max time for B to send message to C
- e : Max time that can elapsed between sprinklers on and the sensors informing B that fire has been put out.

a) Fire engines send for nothing

We're looking here for a situation where a "fire out" is not send on time to the local fire station. Let's assume that two fire start :

1. The first fire start. Message "fire!" is sent from A to B.(time 0)

2. Before B receive the message, a second fire start.(time a, between 0 and d)
3. B receive the fire message, and start the sprinkler.
4. First fire is off. second is still on , so A is still sending message to C. B don't send fire out, as the second fire is still on.
5. C is now receiving message for more 2d+e, so send fire engines.
6. Second fire is off, thanks to sprinklers.
7. B send fire out message, and the time could be between 2d+e and (2d+e) +a.
8. As fire engines are here, no more fire.

There is clearly a system trouble here.

b) Fire engines not send as fire is on

We are looking here for the cases where fire engine are not sent as they should be. It could happens in this case :

1. First fire start, time 0.
2. A send message to B and C. B starts the sprinklers.
3. Sprinklers are on, and achieve to kill the first fire.
4. B send to C a "fire out!"
5. Before C receive the message, a second fire start (a units time before C receive the message).
6. A send message "fire!"to B and C.
7. C receive b message, reboot counter. Then receive message from A.
8. Sprinklers can't kill the second fire.
9. C sent fire engines, but it could 2d+e+a that the fire started. Here fire engines are send in a good purpose, but at the wrong timing. The same problem happens when two fire start, and the second one is kill by the sprinkler before fire engine are send for the first one.

ii) Producing a correct system

We now assume that A and B use logical clocks, and they use those clocks to timestamp their messages. Those clocks increase on each local actions they execute. We also decide that A and B will have the same increment value.
We'll keep the same system architecture, we'll modify A, B and C actions and reactions to messages. Computer A still receive informations from fire detection sensors. each sensors notification increase the A clock value. A raise two events : sending message to B (timestamp x), and sending message to C (timestamp x+i, i the increment of A clock).
Computer B have set his logical clock to the value of the timestamp (let's call it x for the example) received from the "fire!" message from A. If fire put out detection sensors warn B before B receive a new fire message from A : B clocks increment, then B send a message with timestamp x + i, with i the increment of the B clock, to computer C, and then start the sprinkler. Otherwise, it timestamp is updated to the new one send by A.
Computer C receive messages from A : it counter move until reach 2d+e, then send the fire engines. If a message is received from B, C will compare the two messages he received : the last received from A, and the last from B.
If timestamp from B is lower or equal to timestamp from A, C have to keep is counter value. If timestamp from B is higher than timestamp from A, then C reset his counter. AS soon as C counter reach the critical value, fire engines are send.
By using logical for A and B, we can solve problems describe in the previous part.

iii) Condition for correctness with physical clocks

The conditions needed for the system we design before are the folllowing :

- The time between the A sending message to B and A sending message to C must be the same as B getting info from sensors and sending message to C (message from A to C have to be delayed).
- The condition for C must change : If timestamp from B is lower or equal to timestamp+ epsilon from A, C have to keep is counter value. If timestamp from B is higher than timestamp+epsilon from A, then C reset his counter. As soon as C counter reach the critical value, fire engines are send.