NOVID-20 Contagion Spread modeling

Distributed simulation risk estimation

Project goal

Main goal of this project is to estimate the risk of each individual being infected. We do not necessarily aim for probability of infection, but rather an abstract score (e.g. on scale 1-100) of the risk, or of a simple ranking of users (e.g. giving each user the information about which percentile they belong to).

Network data

- **node** - single device (corresponding to a single person) or a single location (such as store, bus, building). Nodes are separated into:
    - **simple_node** - person/device, identified by a unique *node_id* created by that device
    - **complex_node** - location based node identified by a unique node created by the system

- **interaction** - edge occurrence between two nodes in a network. Each interaction is characterized by:
    - **interaction_id** - unique
    - **nodes** interacting
    - **weight** some sort of interaction weight calculated from the total time and/or closeness (depending on what bluetooth can estimate)
    - **time_start** when the interaction started
    - **time_end** when it ended - “moment-interactions” are possible with time_start=time_end

When interaction occurs, each node stores it locally. For simple nodes, this means storing the infection on the device. For complex nodes, we store the interaction on our system (our servers play the part as “complex node devices”). Each node holds all the interactions up until some predefined time threshold (e.g. 14 days). Additionally, users are allowed to manually specify prior interactions.

Data storage

Because privacy is an important issue, storing interaction data in a centralized database is prohibited. Instead, data is distributed across devices - each node holds it's set of neighbors (interactions).

Simulation

Each simulation is started with the prescheduled (periodic) signal broadcasted to all infected nodes from the server containing a unique simulation ID. When received, each infected node initiates a simulation vector of size N (N between 1k and 10k), *s = (1, …, 1)* and sends it over each interaction which occurred after the estimated time of infection (e.g. 5 days before symptoms). If there were multiple interactions since the time of infection with another node, those are all sent separately.

When a node receives such a vector (either from an infected or non-infected node) they randomly set each vector element to 0 with probability p, which depends on the weight of the interaction (higher weight, lower p) - we call this the **pruning** phase. This vector is then stored locally (together with the simulation ID and the interaction ID) and is passed on through all the other interactions which occurred after the interaction through which the vector came (possibly with some additional delay modelling the incubation period) - we call this the **dissemination** phase.

When node receives multiple vectors from different interactions, it performs pruning on each one and then combines them with the OR operation, followed by the dissemination phase - this time sending the vector only through the interactions which occurred after the last of the interactions through which the vectors came in. In other words, if node had interactions on timestamps 1, 2, 3 and 4, and received vectors over interactions 1 and 3, it would disseminate the pruned vector from 1 through interaction 2, and combined vectors (v(1) OR (v(3)) through interaction 4.

This means that it is possible for a single interaction to be used multiple times to spread the vector. If this happens, the receiving node simply discards the previous vector and proceeds as described. This means that the simulation state is getting updated continuously for all nodes, meaning no node can ever know if their current vector would be the final one they receive. This makes it important for the server to broadcast to all the nodes when the simulation has ended - the server can make this call when no new simulation messages are sent for some preset interval of time.

To decrease server load, vectors should be dropped if their total sum falls below some threshold.

Because vectors are marked with simulation IDs, multiple simulations are allowed to overlap in their schedules. This can be useful in cases when message spreading occurs too slow, and the whole simulation takes too long. In that case, we can schedule a new simulation each 6 hours, even if it takes 24 hours for each to finish.

Note that because pruning is modelling the probability of disease getting transferred through an interaction it could also be processed by the sending node, instead of the receiving one. However, it is more probable that these spreading-probability parameters would depend on the characteristics of the receiving node (age for example). However, if these parameters turn out to be highly dependent on both the sender and the receiver, pruning can be done on both sides.

Risk estimation

Once the simulation has ended, each node is left with all the vectors that came in from the interactions. If no vectors were received, risk is low. Received vectors are combined using the OR operator, and final risk is estimated as a total sum of the resulting vector.

Node communication

Communication between each pair of interacting nodes can, if needed, be encrypted, as a centralized server only serves as a messaging medium and never “consumes” those messages. However, IDs of nodes themselves cannot be encrypted in this manner.

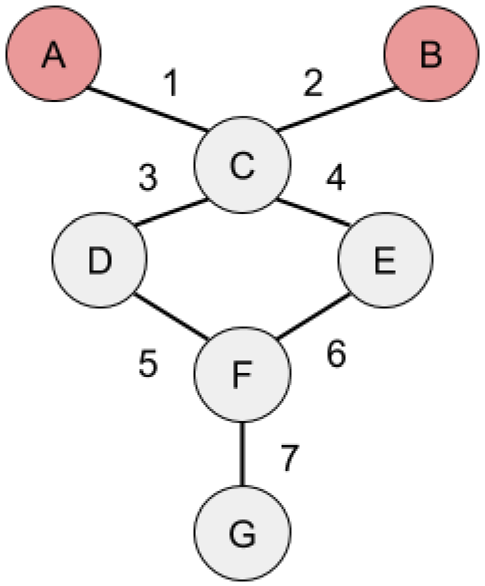
Intuition

Simulation vector can be thought of as N simulation runs in parallel, with each vector element describing one run. Ending up with a vector [0 0 1 1 0] means that this node got infected in 2 out of 5 simulations (in third and fourth one).

Pruning is simply an applied virus propagation probability - for each of the simulation runs (each "element" of the vector), when the virus is "trying" to be transferred over an interaction, we allow this transfer with probability p. Combining multiple vectors with the OR operation simply means "if node is infected from any of its neighbors during the simulation, it is infected", or, in other words, node is not infected only if none of the neighbors transferred the infection.

Example

Consider the following network, with numbers on the edges representing timestamps, and red nodes being infected.



Following is one possible course of action during a simulation:

1. A sends a 1-vector (N=5 for demonstration purposes) [1 1 1 1 1] to C
2. C performs pruning and sends this pruned vector (e.g. [1 0 1 0 0] to B, D and E
3. B sends the 1-vector [1 1 1 1 1] to C
4. C performs pruning on this vector from B, and combines this pruned vector with the previously received vector from A (already pruned) with the OR operator. C then sends this new vector to D and E (not to B because now the latest interaction included in the vectors is interaction 2).
5. D and E have so far received two different vectors from C. They operate in the same manner after each one, pruning and sending them further. These are not combined in any way - previously sent vector from C is dropped, as they were transferred over the same interaction.
6. Etc.

Note that step 2 could have been delayed either because of latency or artificially to reduce the number of repeated messages over the same interaction. In that case, C would combine the vectors and proceed as in step 4, and nodes D and E would only be activated once. Similar situation can then occur with messages coming to F from D and E.

Continuing the example, let's say that D and E received a new vector from C:

- C → D: [0 1 1 1 0] (dissemination)
- C → E:[0 1 1 1 0] (dissemination)

Next, both D and E prune the received vector, leaving each element at 1 with probability p:

- D_p: [0 1 0 0 0] (pruning)
- E_p: [0 0 1 0 0] (pruning)

Node F receives these two vectors:

- D → F: [0 1 0 0 0] (dissemination)
- E → F: [0 0 1 0 0] (dissemination)

Prunes them:

- F1: [0 0 0 0 0] (pruning)
- F2: [0 0 1 0 0] (pruning)

And combines them:

- F1 OR F2: [0 0 1 0 0] (combination)

Finally it can use this vector to estimate its risk, which in this case is 1/5.

Once all the messages are shared and the server broadcasts the "simulation over" message, each node sums the vector they last sent to their neighbors and uses that sum as a risk estimate.

New interactions

When new interaction occurs, there are, by definition, no other interactions for the two involved nodes that occured later than the new one, meaning only the risks of these two nodes need to be updated, and no further propagation is needed. Because each node still holds all the vectors from the last simulation, they can share them and update their risks.