Software Architecture

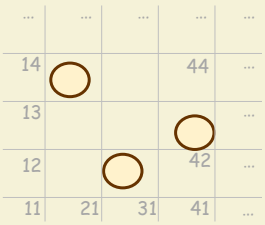
Lecture 7
Project

João Pedro Sousa
George Mason University

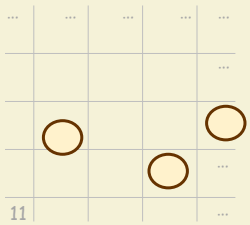
today

- project
  - functional reqs
  - constraints 1
  - constraints 2
  - constraints 3
- QAW
  - availability
  - scalability
  - latency
  - security

context sharing
among mobile devices



zone A



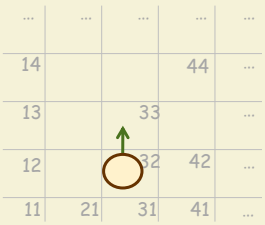
zone B

- each node shares *zero or more* observations for current location
  - fire – likelihood 2%
  - flooded road - 30%
  - looting – 5%
  - supplies available – 10%
  - medvac needed – 5%
- moves within zone
  - likelihood distribution

10	10	10
10	20%	10
10	10	10

stay move east

context sharing
among mobile devices



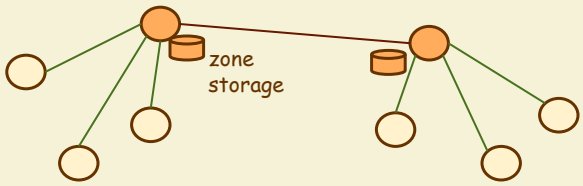
before moving to new location node wants to

- know local observations during last 1 hour
  - fire
  - flooded road
  - looting
  - supplies available
- medvac needed
  - immediately announced to all nodes on both zones

today

- project
  - functional reqs
  - constraints 1
  - constraints 2
  - constraints 3
- QAW
  - availability
  - scalability
  - latency
  - security

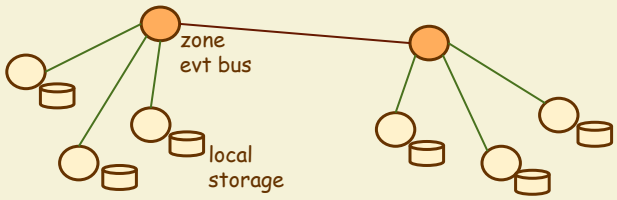
constraint 1
client server - *cloudlets*



- nodes
  - post observations with zone server
  - query zone server before moving
- zone servers
  - store observations (except medvac)
  - push medvac request to all nodes
  - reply to queries

constraint 2

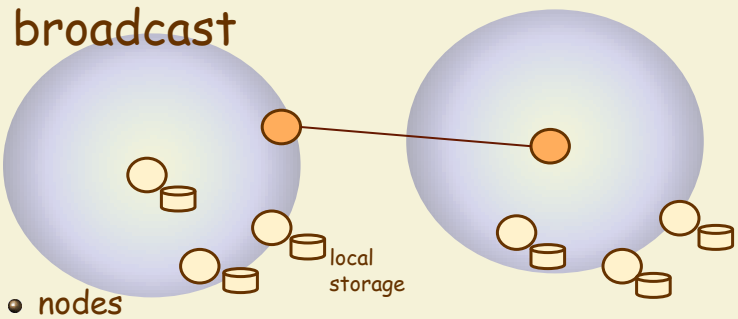
publish-subscribe w/evt bus



- nodes
  - announce observations with zone event bus
  - subscribe to events on *locations* of interest
  - subscribe to medvac
- zone event buses
  - facilitate local pub-sub
  - relay medvac event to other zone(s)

constraint 3

broadcast



- nodes
  - have *no* prior knowledge of other nodes
  - broadcast observations over wireless
  - listen to events
  - remember events on *locations* of interest
- zone relays
  - have previous knowledge of other zone relay nodes
  - relay medvac event to other zone(s)
    over point-to-point connection

take 15

- review
  - QAW
  - QA guidelines
    - availability
    - scalability
    - latency
    - security

example scenario
availability – priority 1

stimuli
- needs obs on loc b

responses
- obs about loc b

artifact
- all running nodes in zone X

source
- any node in zone X

environment
- 100 nodes
- 2 zones
- 900 blocks in X
- 2 minutes between moves

response measures
- 95% chance get some obs
- 70% complete obs

example scenario
availability - priority 1

stimuli
- request medvac

responses
- all nodes in all zones receive it

artifact
- all running nodes in all zones

source
- any node in zone X

environment
- 100 nodes on each zone
- 2 zones
- 900 blocks in X
- 2 minutes between moves

response measures
- at least 80% nodes receive it 70% of the time

example scenario
scalability - priority 2

stimuli
- add 1 node to zone X

responses
- node becomes part of the system in zone X

artifact
- all running nodes in zone X

source
- system supervisor

environment
- 100 nodes in zone X
- 2 zones
- 900 blocks in X
- 2 minutes between moves

response measures
- <5 seconds to add each node
- always succeeds up to 500 nodes in a zone

example scenario
scalability – priority 2

stimuli
- report obs

responses
- obs become available to others

artifact
- all running nodes in zone X

source
- any node n

environment
- 100 nodes in zone X
- 2 zones
- 900 blocks in X

response measures
- accommodate rates up to 10 obs per minute

example scenario
scalability – priority 3

stimuli
- request obs on block b

responses
- obs is delivered

artifact
- all running nodes in zone X

source
- any node n

environment
- 100 nodes in zone X
- 2 zones
- 900 blocks in X

response measures
- accommodate rates up to 10 req per minute

example scenario
response time - priority 2

stimuli
- req obs

responses
- get obs

artifact
- all running nodes in zone X

source
- any node n

environment
- 100 nodes in zone X
- 2 zones
- 900 blocks in X
- 1 minute between moves

response measures
- 5s +/- 2s

example scenario
response time - priority 3

stimuli
- request medvac

responses
- reception by every node

artifact
- all running nodes in all zones

source
- any node n

environment
- 100 nodes on each zone
- 2 zones
- 900 blocks in X
- 1 minute between moves

response measures
- 10s +/- 5s

example scenario
security – priority 10

stimuli
- tries to join zone X

source
- untrusted node

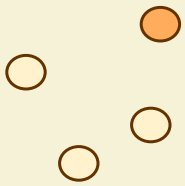
artifact
- all running nodes in zone X

environment
- 100 nodes
- 2 zones
- 900 blocks in X

responses
- reject node
- log attempt

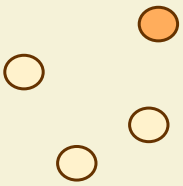
response measures
- 100% detection of untrusted nodes

experiment setup – 1 zone

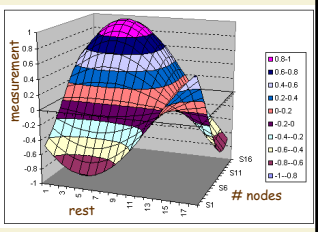


- start each node/server as a separate process on hydra
  - use your favorite technique for *automated* bootstrap, e.g. a script
- all communications via the TCP stack: RMI, p2p messages, multicast…
- measure
  - m 1: latency of *get obs*
  - m 2: completeness of *obs*
    (# records returned by get obs / # postings for the location)
  - m 3: latency of *adding* one node
  - m 4: latency between *medvac* announcement and reception

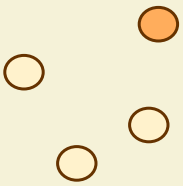
experiments



- 6x5=30 experiments
  - vary # nodes per zone: 10, 20, 50, 100, 200, 500
  - vary rest between moves: 1, 6, 12, 60, 120 seconds
  - goal: obtain data points to plot surface graphs
- for each experiment
  each node makes 30 moves and terminates
  expected experiment duration = 30 x rest, i.e. up to 1h



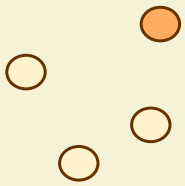
experiment load



- simulate behavior of each node
  - use a log-normal distribution to simulate the time between moves: http://en.wikipedia.org/wiki/Log-normal_distribution
  - e.g. for an average rest of 6 seconds set a random sleep time as follows:

```
Random x = new Random(System.currentTimeMillis());
  <...>
  double seconds = Math.exp(x.nextGaussian()/2 + Math.log(6));
  Thread.sleep((long) (1000*y))
  <...>
```

experiment load



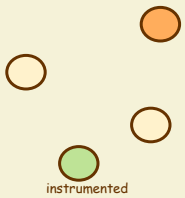
- when timer elapses
  - toss uniform coins for each kind of observation
  - toss uniform coin to determine move

10	10	10
10	20%	10
10	10	10

stay move east

```
Random x = new Random(System.currentTimeMillis());
  <...>
  boolean fire = (x.nextFloat() < 0.05);
  <...>
  float m = x.nextFloat();
  boolean stay = (m<0.20);
  boolean move_east = (0.20<=m && m<0.30);
  <...>
```

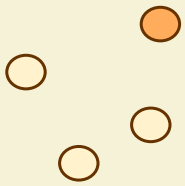

gathering data



- the bootstrap script launches
  - one zone server (c1 and c2 only)
  - n-1 nodes
  - 1 instrumented node *n*

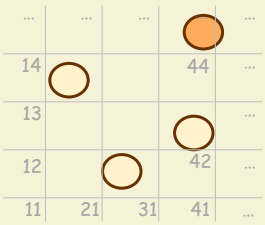
- the scrip registers time to create node n-1 and node *n*
  - produces average latency to create a node : one data point for m3
- during moves 5 to 15 (steady state of the system) node *n*
  - registers *get obs* latency
    - before terminating produces average latency: one data point for m1
  - registers number of obs returned by *get obs* for each location
    - calculate ratio to complete set of obs for each location (known by system) to produce average completeness : one data point for m2

gathering data



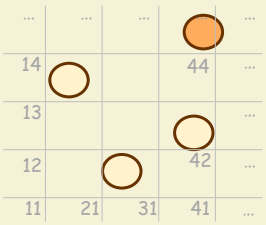
- all medvac requests are *time-stamped*
- *every node* logs medvac latency (currentT – timeStamp) to a file
- after all nodes terminate, run a script that scans all log files to compute average medvac latency: one data point for m4
  - register size of log file (# issued medvacs) for each experiment

memory footprint



- during system steady state, measure
  - memory footprint of zone server (c1, c2)
  - memory footprint for one node (c1, c2, c3)

completeness-memory tradeoff
(c2 and c3)



- previous experiments assumed each node views entire 30x30 zone
- run additional m2 experiments with 50 nodes and 12s avg rest
  - varying view window: 3x3, 5x5, 10x10 (already have 30x30)
  - as a function of view size
    - plot curve of completeness
    - plot curve of memory footprint for one node

time permitting: *demo* setup

- nodes
  - zone A: at least 2 separate machines supporting 3 or more nodes
  - zone B: at least 1 machine supporting 3 or more nodes
- 2 zone servers
  - each on a separate machine
- (total min 3 separate machines)