MetroTrack
Presented By Philip Shibly

Predictive Tracking of Mobile Events Using Mobile Phones

Gahng-Seop Ahn, Mirco Musolesi, Hong Lu, Reza Olfati-Saber, and Andrew T. Campbell, “Metro Track: Predictive Tracking of Mobile Events Using Mobile Phones.”The City University of New York, USA, gahn@ccny.cuny.edu University of St. Andrews, United Kingdom Dartmouth College, Hanover, NH, USA

Contents

What is MetroTrack

- Mobile Phone Event Tracking System
- Tracks moving targets by collaborative sensing devices.
- Predicts future location of a target that may be lost during tracking.
- Does not rely on static networks or backend computation, but rather mobile users, so it is susceptible to spacial density, user participation, and realtime computation/feedback needs.

Initial Pros/Cons

- Relies on participatory users.
- Does not rely on user action.
- Needs dense network of users.
- Why would someone participate to track someone else's target? Why participate at all?
- Mobility of users is unpredictable and uncontrollable.
- Requires common sensors between users.
- Requires application to be running.
- Battery consumption?
- Does not rely on central nodes.
- Does not use node grouping.
- No back-end requirements.

Framework

- MetroTrack consists of two algorithms
- (1) Information-Driven Tracking
  - The sensor node begins tracking when certain criteria is met.
  - Forwards tracking task to neighboring nodes.

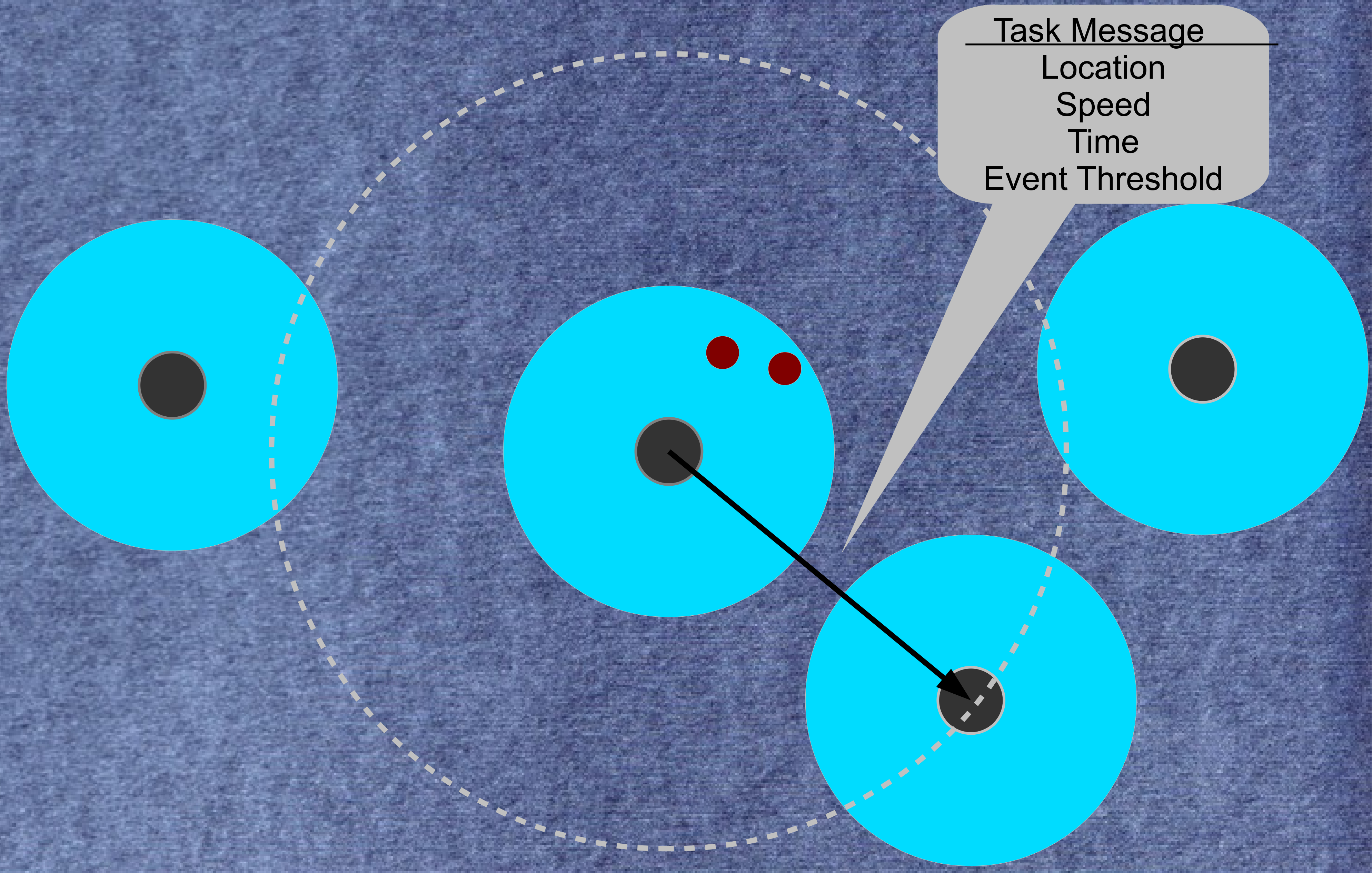
Framework (cont.)

- (2) Prediction-Based Recovery
  - If a nodes neighbor(s) do not report a tracked event task, then it is assumed that the target is lost.
  - Recovery is based on estimation of the targets location and the margin of error associated with the prediction.

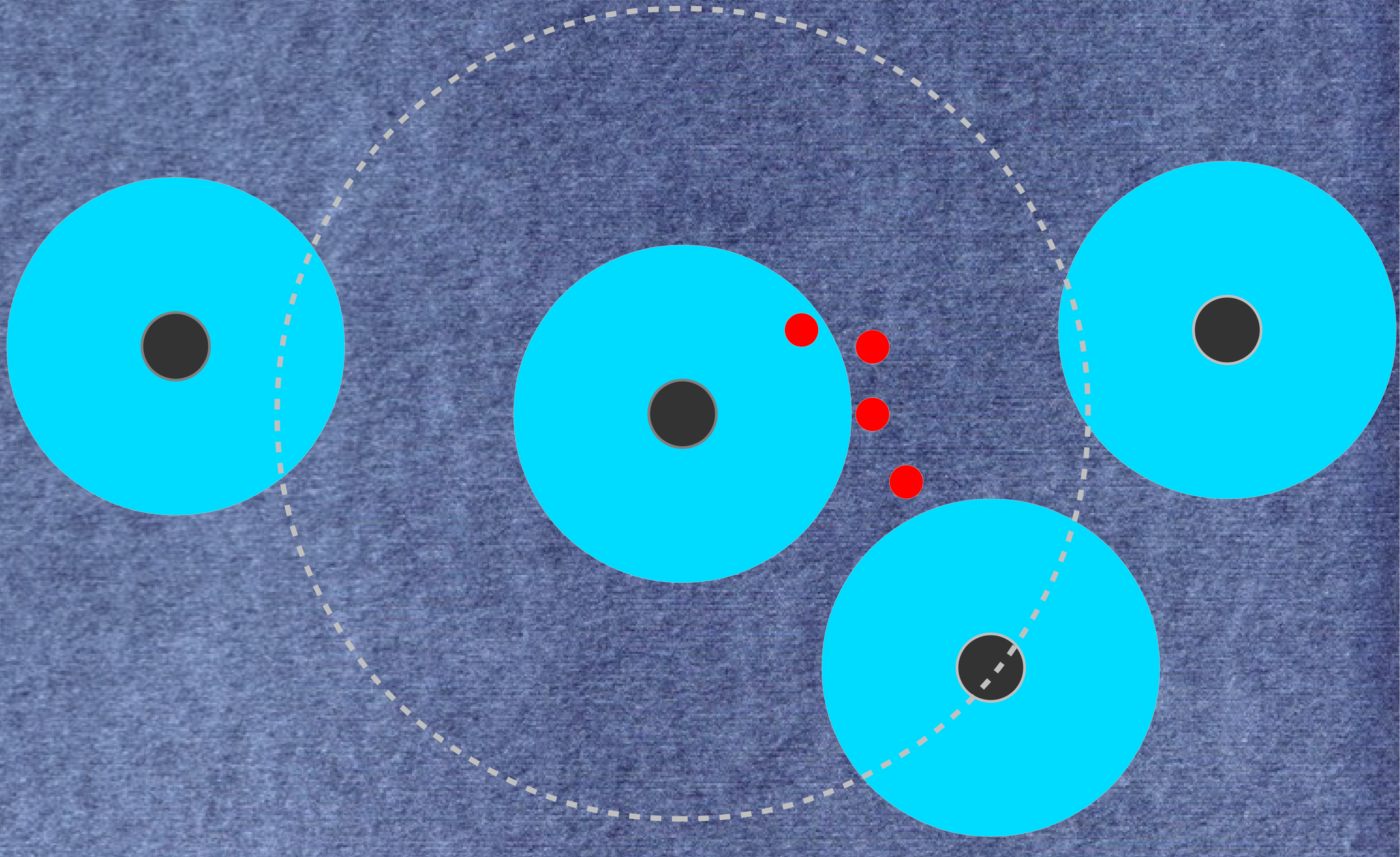
Information-Driven Tracking

- User Initiation or Sentry Node Initiation detects a target.
- The node sends a task message to its first nearest neighbors (one hop away).
- A node only forwards a task message if it has received a task message AND detects the target.
- A node that has detected a target AND sends a task message listens for the same task message to be sent back.
  - If it does not receive a task message back from any of its neighbors, it assumes the target is lost.

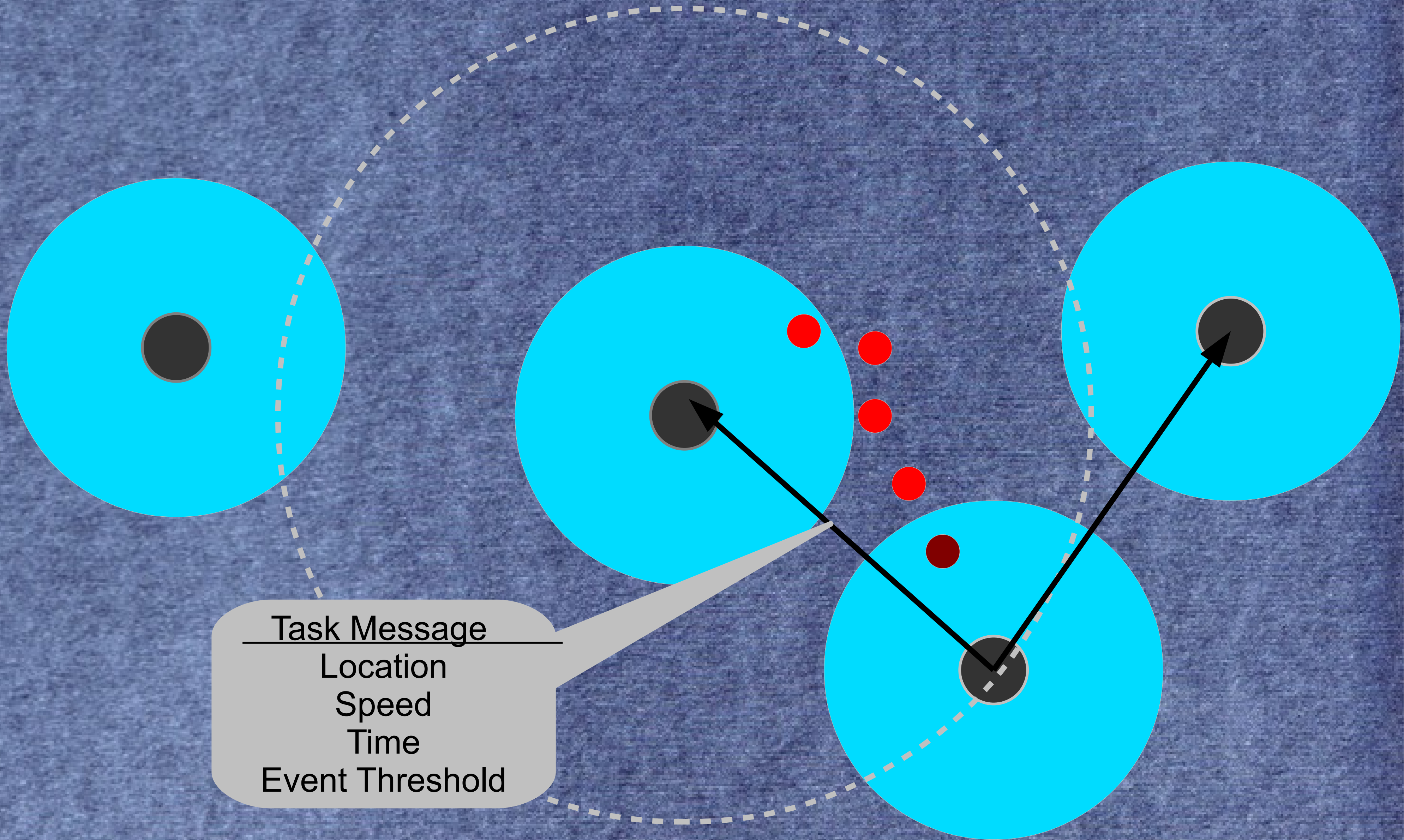
Information-Driven Tracking



Information-Driven Tracking



Information-Driven Tracking



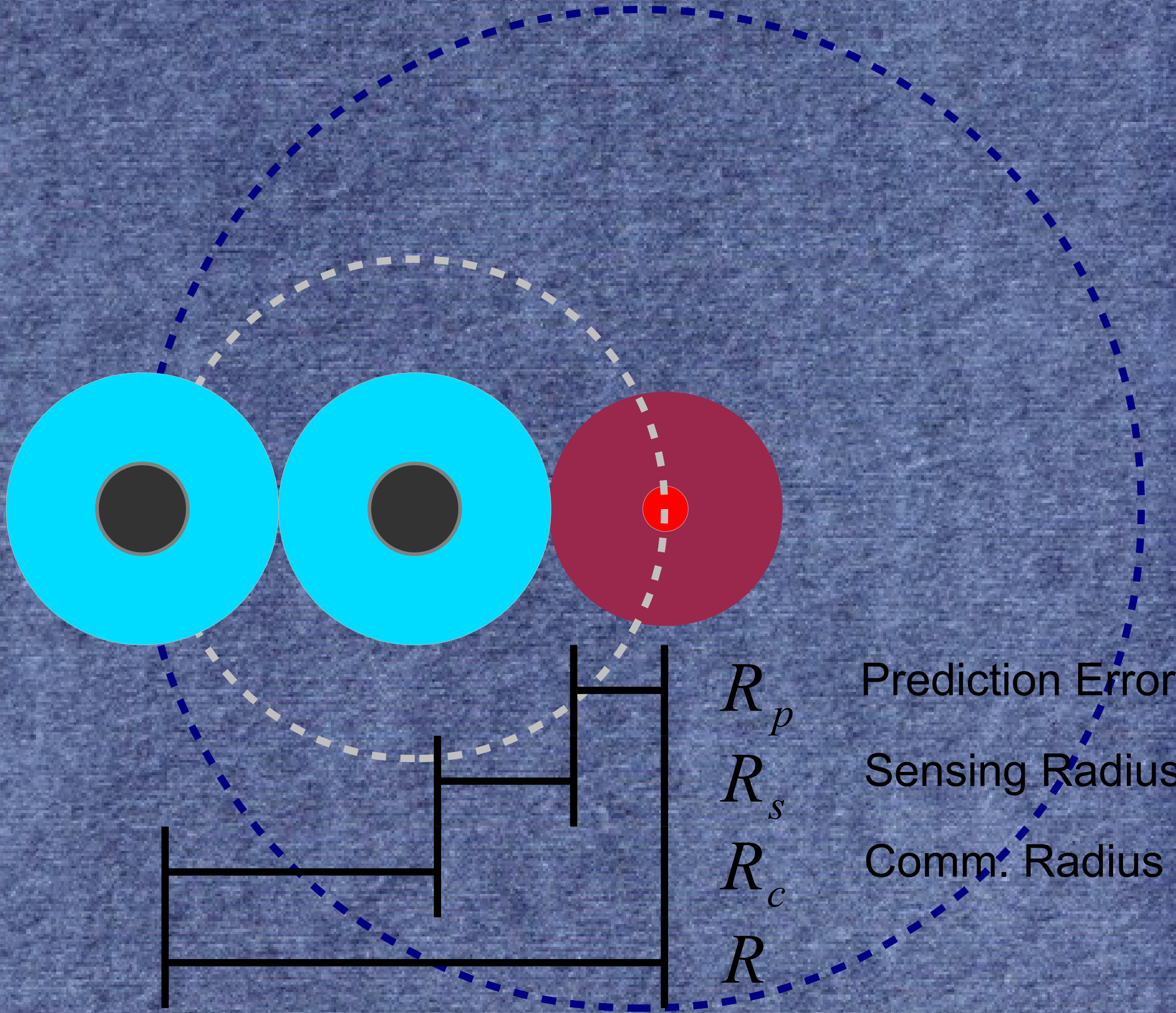
Prediction-Based Recovery

- A target is not lost just because a sensor is not detecting it.
- Only if a sensor does not receive a responding task message from a neighboring node, does it infer that the target is lost.
- Once a node determines that a target is lost, it broadcasts a "recovery message" that is projected to nodes in the location to which the target is likely to move.
  - This is done by a geocast scheme.
- If a node that receives the recovery message detects the target, then recovery is complete.
  - A task message is then sent out, and MetroTrack begins again at the Information-Driven Tracking phase.

Prediction-Based Recovery

- Determining the Projection Location:

$$R = R_p + R_s + R_c$$



Prediction-Based Recovery

- A node that receives the recovery message stays in the recovery state until it moves outside the Projection Location or a recovery timer expires.
- Once the recovery timer expires and the target is not recovered, then MetroTrack stops tracking the target.
- If the target is recovered, the recovery node broadcasts a suppression message so other nodes know to quit sending the recovery message and end the recovery process.

Algorithm Assumptions

- In order to develop the prediction algorithm for the recovery process, the authors make the following assumptions:
  - The target velocity is comparable to the node velocities.
  - Node sampling rate is high enough to detect the target at the targets given velocity.
  - The targets velocity is represented as a Constant Velocity model (dynamically changing velocity with constant known variance).
  - The target thresholds are unique to that target.

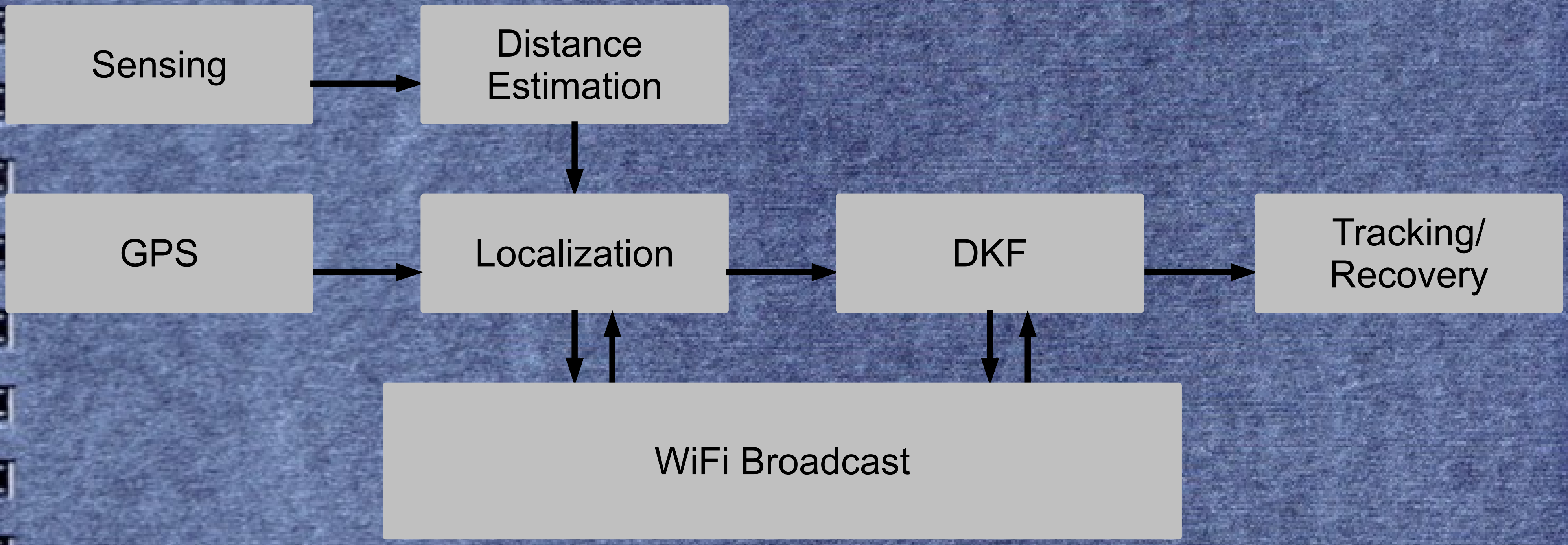
Prediction Algorithm

- Kalman Filter is used as a predictor so the process is represented as a linear state estimator and the process noise is represented as zero-mean Gaussian white noise. The measurement noise is associated with the prediction of the next step, k.
- The variance of the measurement noise dictates the center and radius of the Projection Location.
- The target can then be assumed to be within the Projection Location with approximately 95% accuracy.

Next Step: Distributed Kalman-Consensus Filter (DKF)

- Each node runs the Prediction Algorithm (Kalman Filter) with their own locally aggrigated data and covariance matrices.
- Then the state estimates (position and velocity) of the target are updated.

DKF Architecture



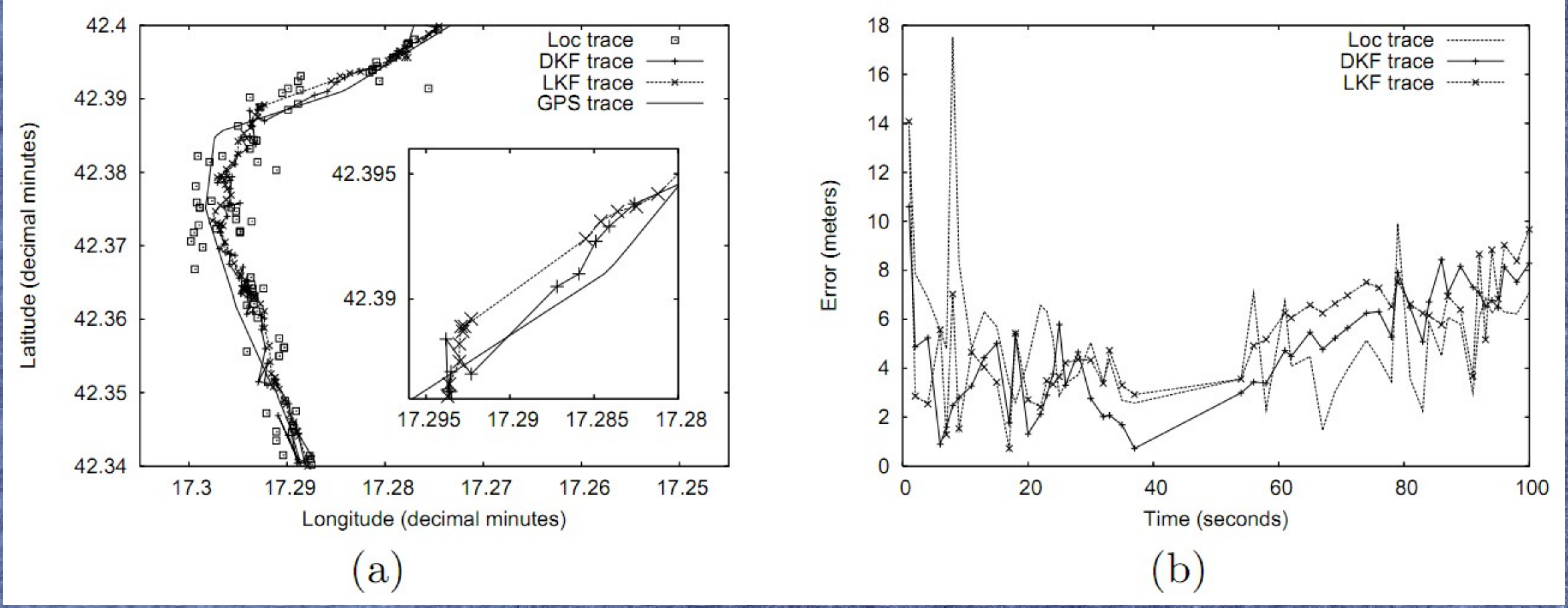
Experiment

- Implemented a Local Kalman Filter and the Distributed Kalman Filter as the prediction mechanisms.
  - Local Kalman Filter does not implement information sharing between nodes.
- Testbed consisted of Nokia smartphones and a bike with a boombox on it.
- The goal was to track the bike with the music playing.

Experiment

- Constant Pink noise was played
- Bike moved at walking speed
- Sound was sampled for 0.5 seconds every 2 seconds by 11 phones
- WiFi transmission with a communication range of 25-30 meters.
- Localization (trilateration) is used to calculate the location of the target
- Users allowed to move around within 40m of the target and randomly in-and-out of the sensing range of 20m.

Results



Sound was turned off from 37s to 54s to emulate a lost target

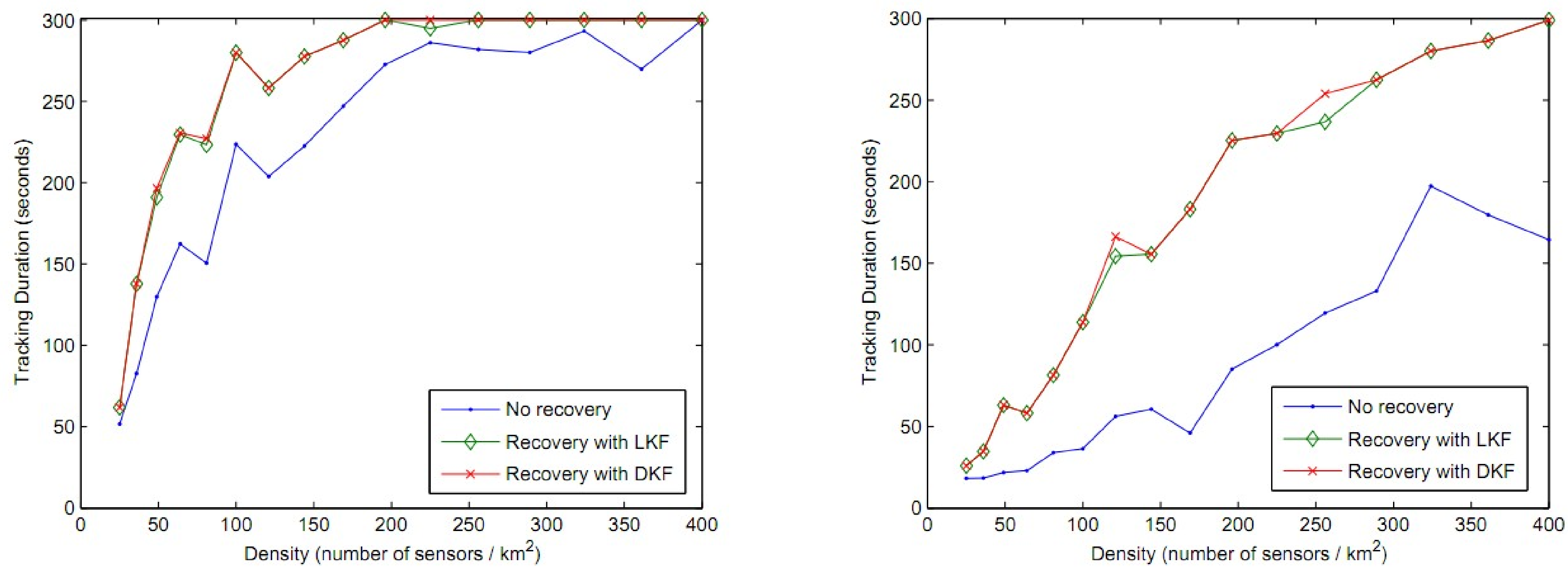
Experiment Critique

- Noisy localization said to be caused by noisy RMS estimation of the sound signal and also GPS positioning error.
  - Hard to consider the effectiveness of the MetroTrack algorithm when an experiment is chosen that introduces its own variance and error.
  - Standard Deviation of the sound source localization error for DKF must be found by trial-and-error before use (used for covariance matrix in Kalman Filter). Applicable to use realtime?
- Localization requires sharing of data between users. Is privacy a concern for the users? It can be compromised.
- Localization by trilateration relies on knowing the original volume of the target sound and the pattern of the sound attenuation over distance (environment affects this as well).
- Sound source is omni-directional, but in realtime would users know if it the target sound is truly omni-directional or would it miss the target if it was within the sensing range but behind the target sound?
- If sound was only turned off for 16 seconds, the bike would only have traveled a max of 7 meters.... hardly a viable distance when there is 11 sensors within a 40 meter radius from the bike at all times, all having a communication range of 25 meters.

Matlab Simulation

- Simulated multiple deployment scenarios with each scenario run 20 times for 300 seconds each.
- Simulation area is 1000m x 1000m
- Target transmission range of 100m
- Sensing ranges of 50m and 100m tested
- Timeout for recovery process is 20 seconds
- Objectives:
  - Track the target for as long as possible without losing it.
  - Track the target as long as possible using recovery with DKF and LKF.

Results



(a) Sensing range of 100 m.   (b) Sensing range of 50 m.

Hint: Notice the x-axis values

Questions?