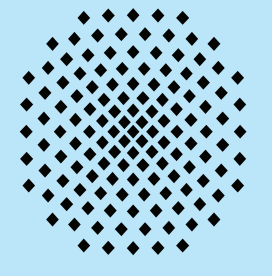


Efficient Tracking of Moving Objects using Generic Remote Trajectory Simplification



Universität Stuttgart

Ralph Lange Frank Dürr Kurt Rothermel
Institute of Parallel and Distributed Systems

SFB 627 **NE**XUS

Motivation

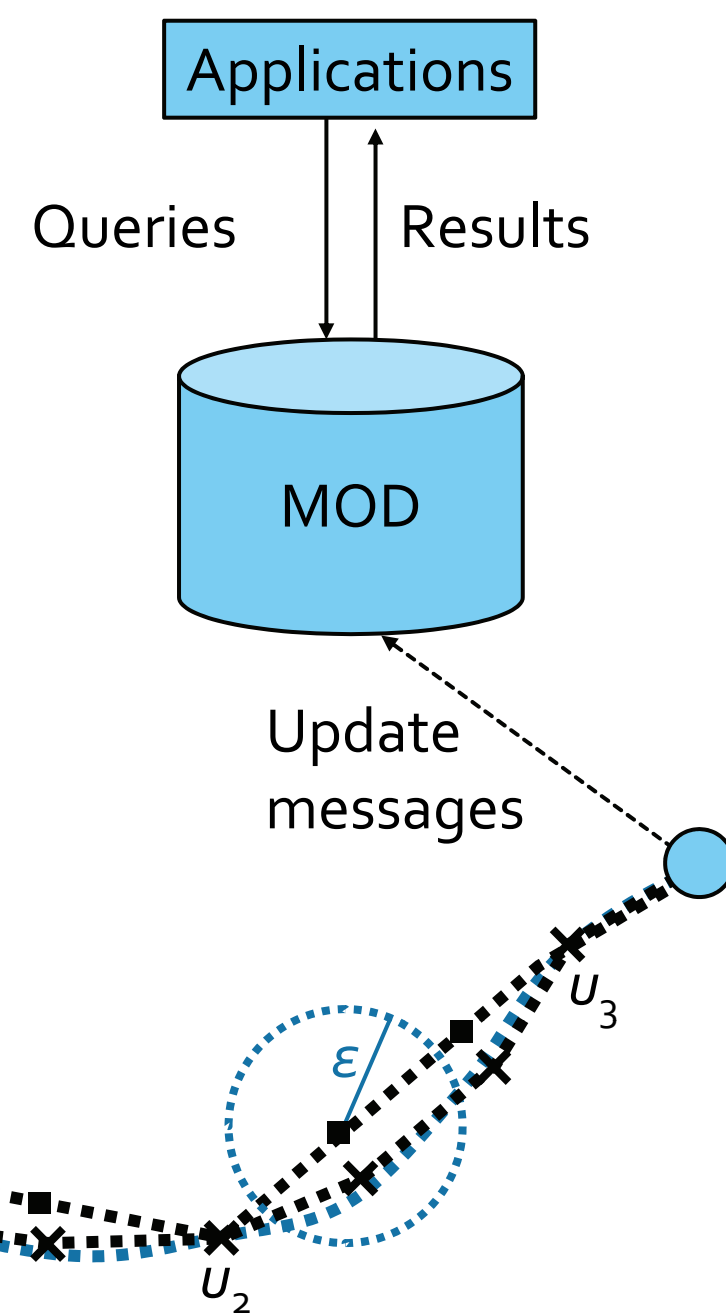
Applications refer to **past** and **present** positions

- Many examples in logistics, transport, sports, ...
- Requires real-time trajectory management

Problem: Large amounts of trajectory data

- GPS receiver generates $3 \cdot 10^7$ records per year
- High communication cost
- Consumes a lot of storage capacity
- High costs for query processing

→ How to reduce trajectory data on **moving objects in real-time** ?



Formal Problem Statement

Kinds of trajectories

- Actual: $\mathbf{a}(t)$ is function $\mathbb{R} \rightarrow \mathbb{R}^d$
- Sensed: $\mathbf{s}(t)$ with vertices s_{1t}, s_{2t}, \dots
 - Attribute $s_i.p$ denotes position at time $s_i.t$
- Simplified: $\mathbf{u}(t)$ with vertices u_{1t}, u_{2t}, \dots

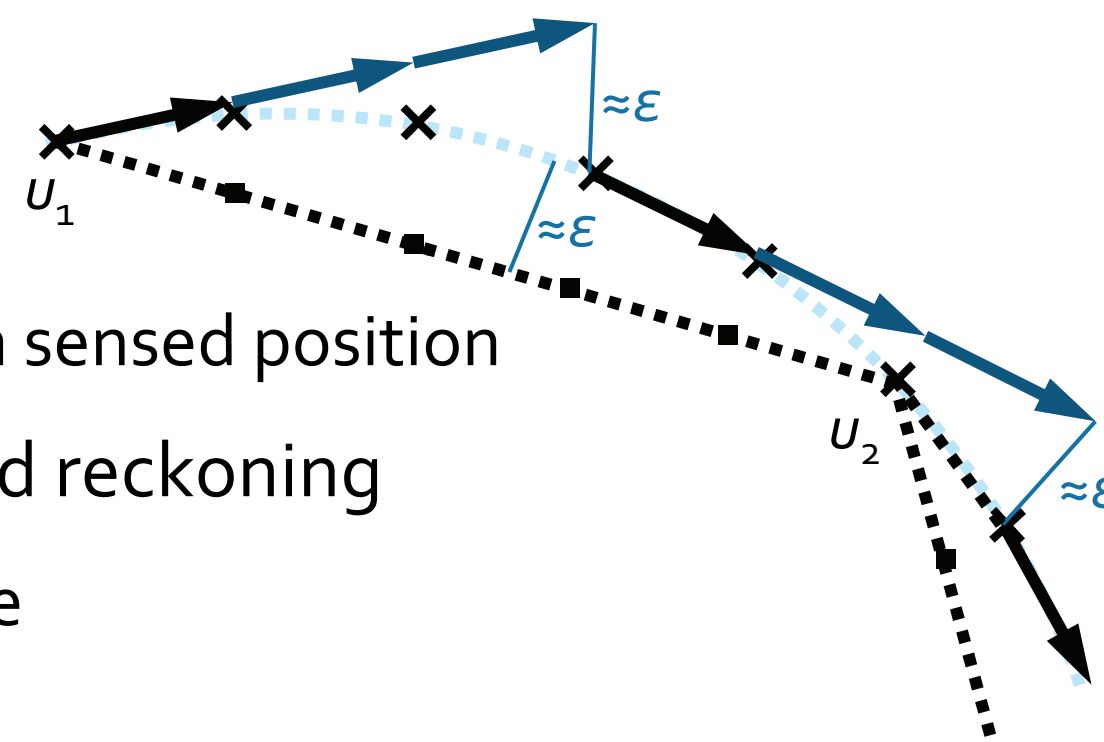
Remote Trajectory Simplification (**RTS**)

- Optimize $|(u_{1t}, u_{2t}, \dots)|$ and communication cost
- Simplification constraint:** $|\mathbf{u}(t) - \mathbf{a}(t)| \leq \epsilon \forall t$
- Real-time constraint:** At current time t_C , $\mathbf{u}(t)$ is available at MOD for $t \in [s_1.t, t_C]$

Generic Remote Trajectory Simplification

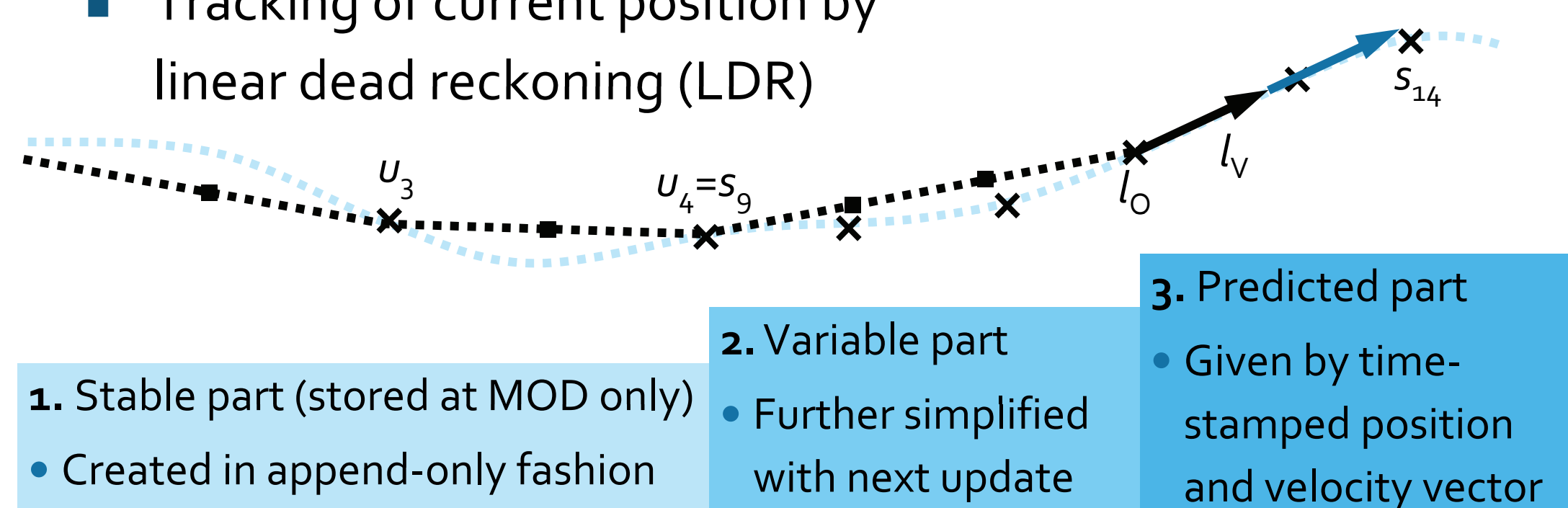
Related approaches

- Line simplification at MOD
 - Requires transmitting each sensed position
- Simplification based on dead reckoning
 - Bad reduction performance



GRTS separates

- Simplification of past movement **from**
- Tracking of current position by linear dead reckoning (LDR)



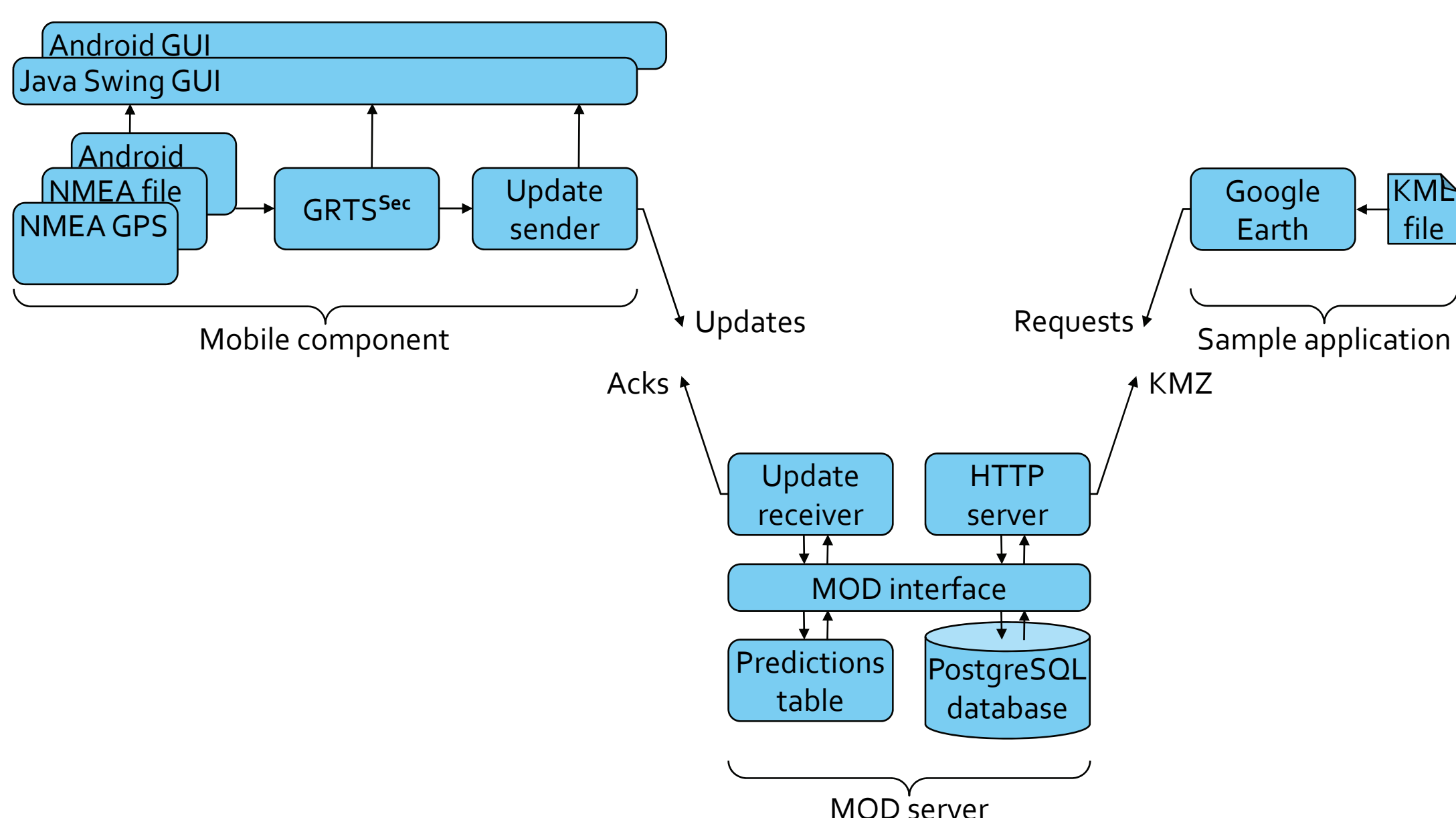
GRTS algorithm

- Executed at moving object
- History of sensed positions of variable and predicted part
 - Bounded size to limit computing time
 - Enhanced with internal compression approach
- Allows for different line simplification algorithms
 - Computational cost vs. reduction efficiency

```

S ← () // Sensing history
...
V ← () // Variable part
U ← () // Vertices for update
while true do
  s_C ← sense position
  S ← S || s_C
  if |S| = m then
    U' ← simplify S with bound ε - δ
    U' ← U' \ (first(U'))
    U ← U || (first(U'))
    S ← (s_i ∈ S : s_i.t ≥ last(U).t)
  end if
  if LDR causes update then
    U' ← simplify S with bound ε - δ
    U' ← U' \ (first(U'))
    U ← U || U'
    l_V ← new velocity prediction ...
    send update (|V \ U|, U \ V, l_V)
    V ← U'
    U ← ()
  end if
end while
    
```

Prototypical Implementation



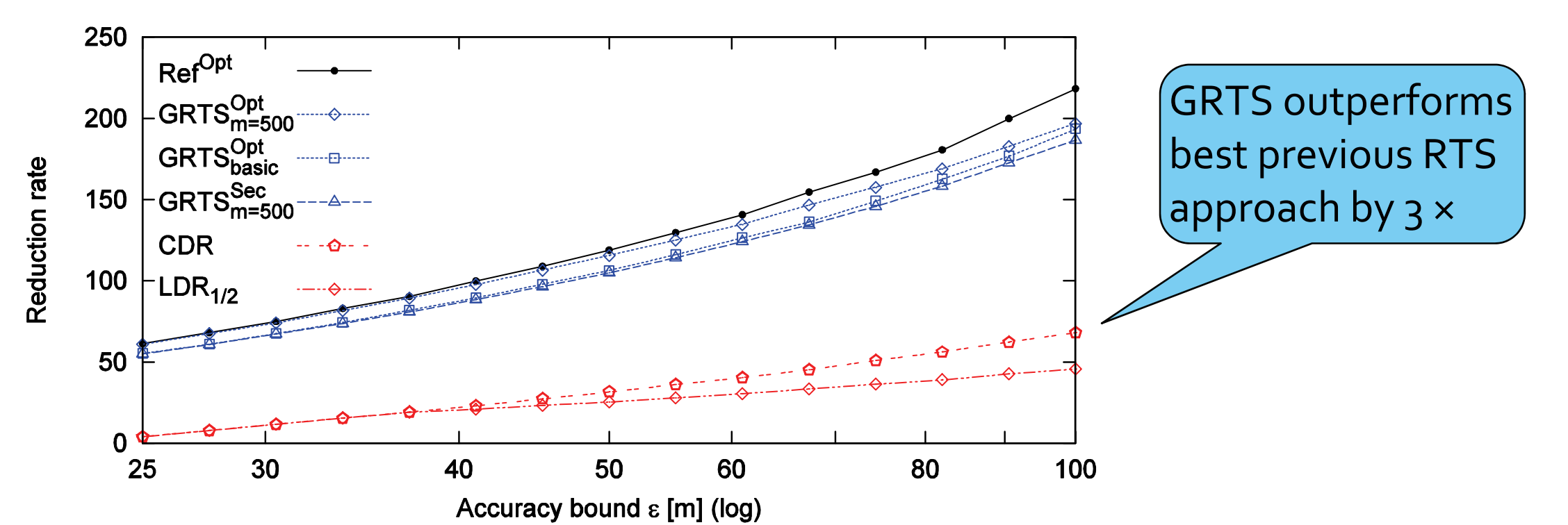
Implementation details

- OpenStreetMap for visualization at moving objects
- Clock synchronization between moving objects and MOD
 - Updates include sent time and previous round trip times
- Section Heuristic (Sec) for line simplification

Evaluation Results

Comparing GRTS to other RTS and offline algorithms

- Simulated with **> 400 h** GPS traces from OpenStreetMap



GRTS outperforms best previous RTS approach by 3x

