

# Dead Reckoning

- Online Multiplayer Games
  - Multi players/ computer connected via Internet
  - Quake, Half Life, etc.
- Characteristics
  - Player moving around rapidly
  - Powerful computers
  - Limited networks capacity

# Dead Reckoning

- Desired features
  - Minimize network traffic
  - Accurately portray others players' movement.
- Dilemma
  - Without frequent updates the local model does not know the current position of remote players

# Dead Reckoning

- BASIC DEAD RECKONING MODEL (DRM)
  - Approximate current position based on past observations (ship sailing in the ocean)
  - Generating State updates;
  - Position extrapolation.
- REFINEMENTS
  - Time compensation:
  - Smoothing

# Modeling

- Each player is an atomic DVE
  - moves at regular intervals independently of others players;
  - receives (external) updates about other players' position;
  - Positions of the others players (display) updated at the rate of 60 times for second;
  - each player maintains locally the position of others players and every 17 milliseconds ( $1/60$ ) generates a suitable graphical image.

# Modeling

- Each player
  - broadcast his current position to the other players every 17 milliseconds;
  - Position information corresponds to location when the message was sent;
  - doesn't take into account delays in sending message over the network

## Limitations

- Position information corresponds to location when the message was sent; doesn't take into account delays in sending message over the network
- Requires generating many messages if there are many vehicles
  - Example:
    - 100 players
    - 0.1 milliseconds to process a single message

## Modeling

- Each player is an atomic DVE
  - moves at regular intervals independently of others players;
  - receives (external) updates about other players' position at undetermined times.
- Each player 'holds' a local Dead Reckoning Model of others players.

## Conclusion

- Dead Reckoning = Approximation

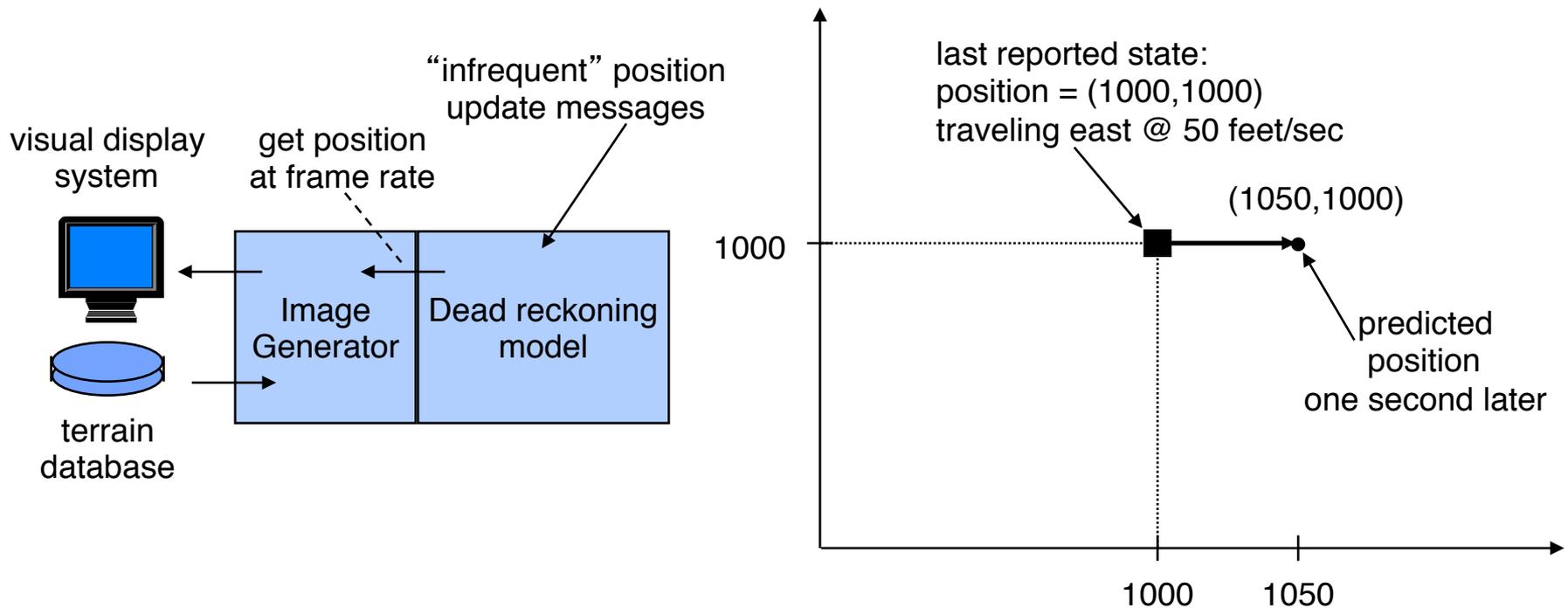
-Advantages:

Reduced network traffic;

Dead Reckoning is used in almost every distribute environments.

# Dead Reckoning

- Send position update messages less frequently
- local dead reckoning model predicts the position of remote entities between updates

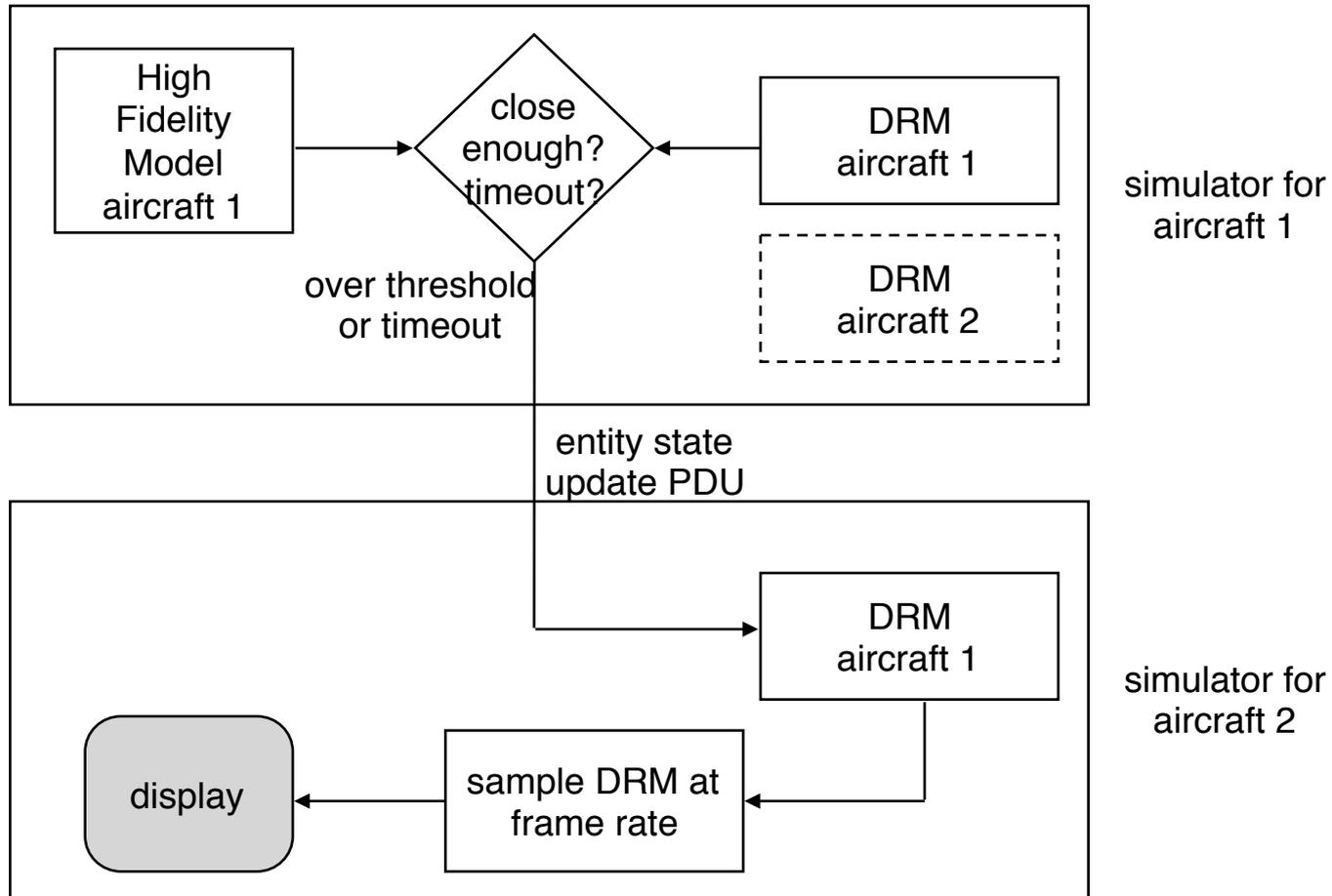


- When are updates sent?
- How does the DRM predict vehicle position?

# Re-synchronizing the DRM

When are position update messages generated?

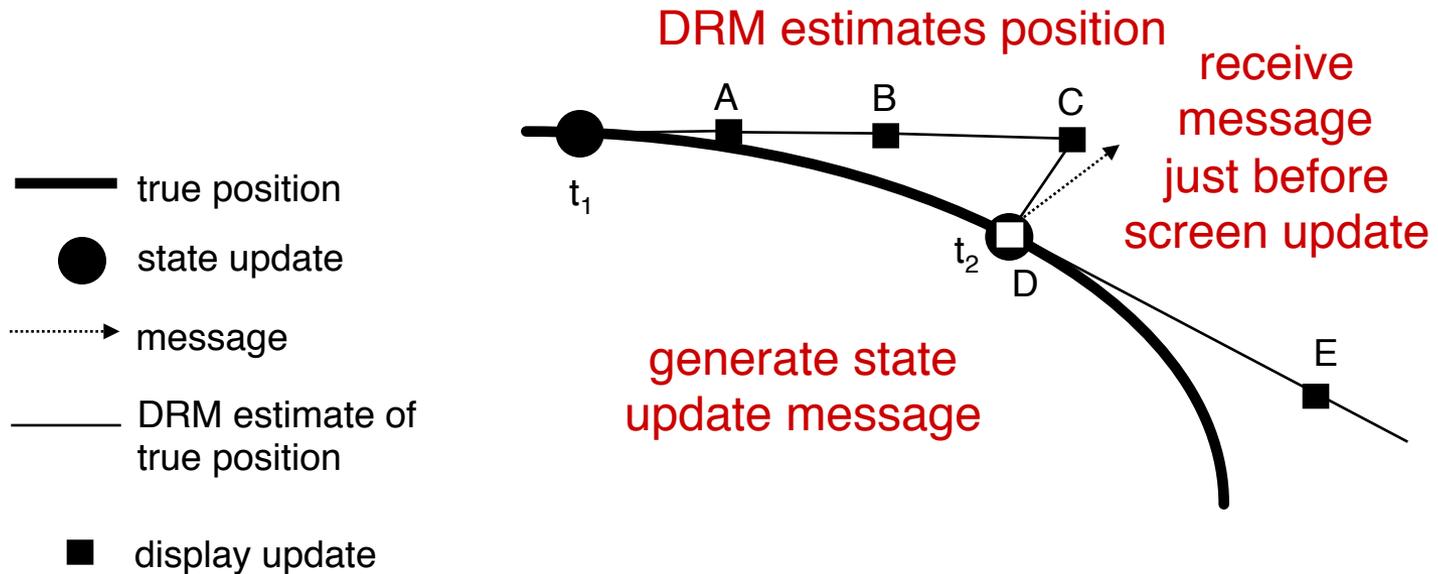
- Compare DRM position with exact position, and generate an update message if error is too large
- Generate updates at some minimum rate, e.g., 5 seconds.



## Dead Reckoning Models

- $P(t)$  = precise position of entity at time  $t$
- Position update messages:  $P(t_1), P(t_2), P(t_3) \dots$
- $v(t_i), a(t_i)$  =  $i$ th velocity, acceleration update
- DRM: estimate  $D(t)$ , position at time  $t$ 
  - $t_i$  = time of last update preceding  $t$
  - $\Delta t = t_i - t$
- Zeroth order DRM:
  - $D(t) = P(t_i)$
- First order DRM:
  - $D(t) = P(t_i) + v(t_i) * \Delta t$
- Second order DRM:
  - $D(t) = P(t_i) + v(t_i) * \Delta t + 0.5 * a(t_i) * (\Delta t)^2$

# DRM Example



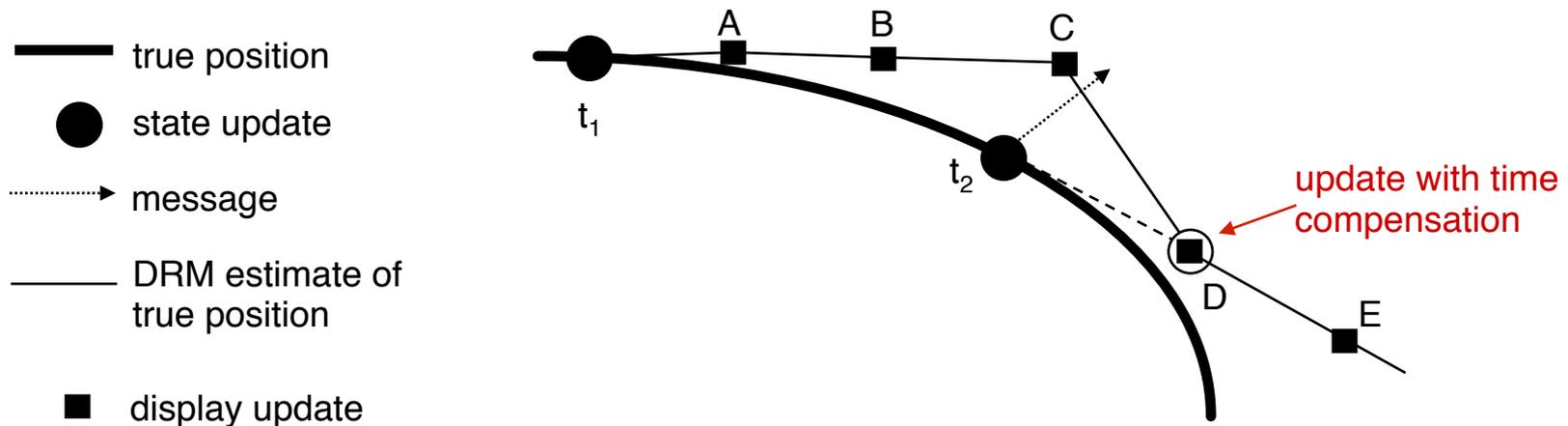
## Potential problems:

- Discontinuity may occur when position update arrives; may produce “jumps” in display
- Does not take into account message latency

# Time Compensation

Taking into account message latency

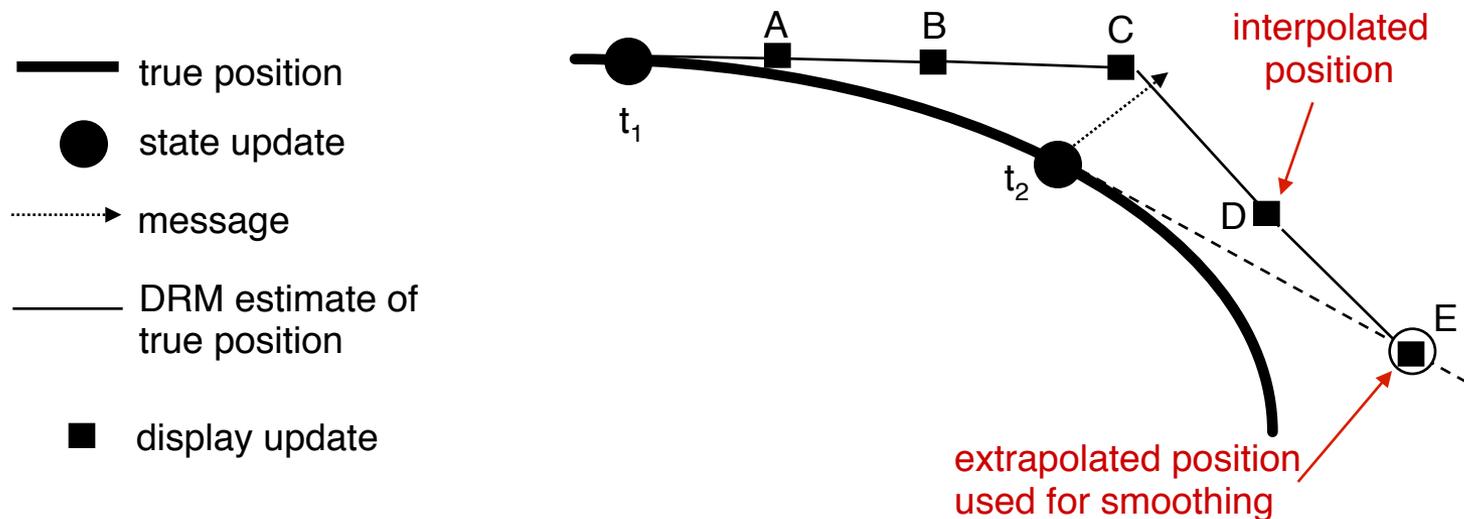
- Add time stamp to message when update is generated (sender time stamp)
- Dead reckon based on message time stamp



# Smoothing

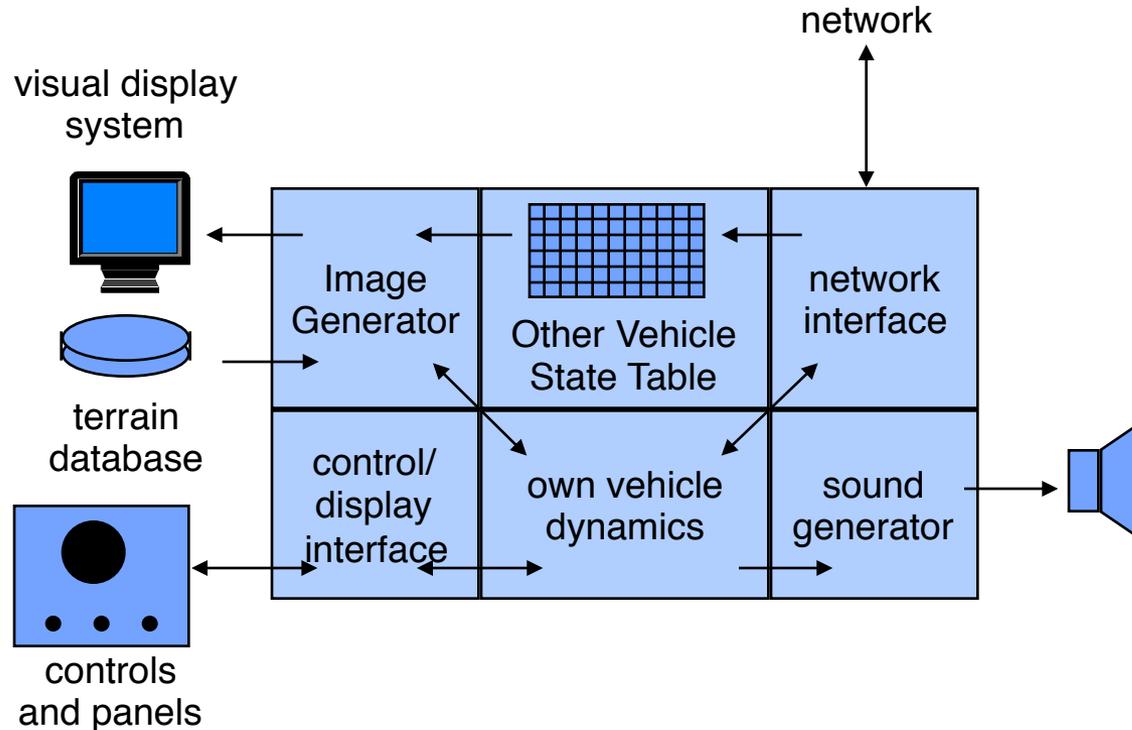
Reduce discontinuities after updates occur

- “phase in” position updates
- After update arrives
  - Use DRM to project next k positions
  - Interpolate position of next update



Accuracy is reduced to create a more natural display

# A Typical DVE Node Simulator



Execute every 1/30th of a second:

- receive incoming messages & user inputs, update state of remote vehicles
- update local display
- for each local vehicle
  - compute (integrate) new state over current time period
  - send messages (e.g., broadcast) indicating new state

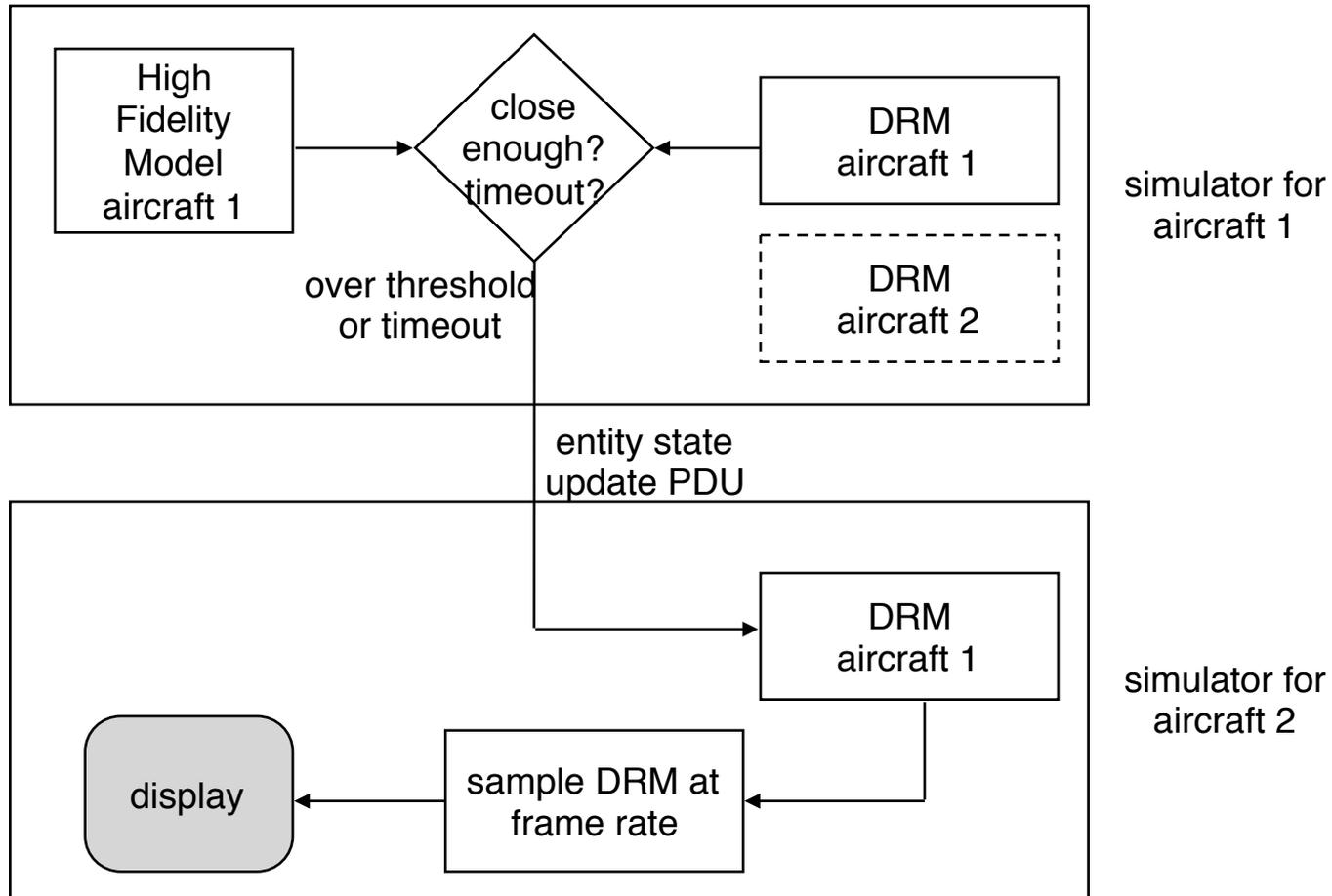
## Distributed Simulation Example

- Virtual environment simulation containing two moving vehicles
- One vehicle per federate (simulator)
- Each vehicle simulator must track location of other vehicle and produce local display (as seen from the local vehicle)
- Approach 1: Every 1/60th of a second:
  - Each vehicle sends a message to other vehicle indicating its current position
  - Each vehicle receives message from other vehicle, updates its local display

# Re-synchronizing the DRM

When are position update messages generated?

- Compare DRM position with exact position, and generate an update message if error is too large
- Generate updates at some minimum rate, e.g., 5 seconds (heart beats)

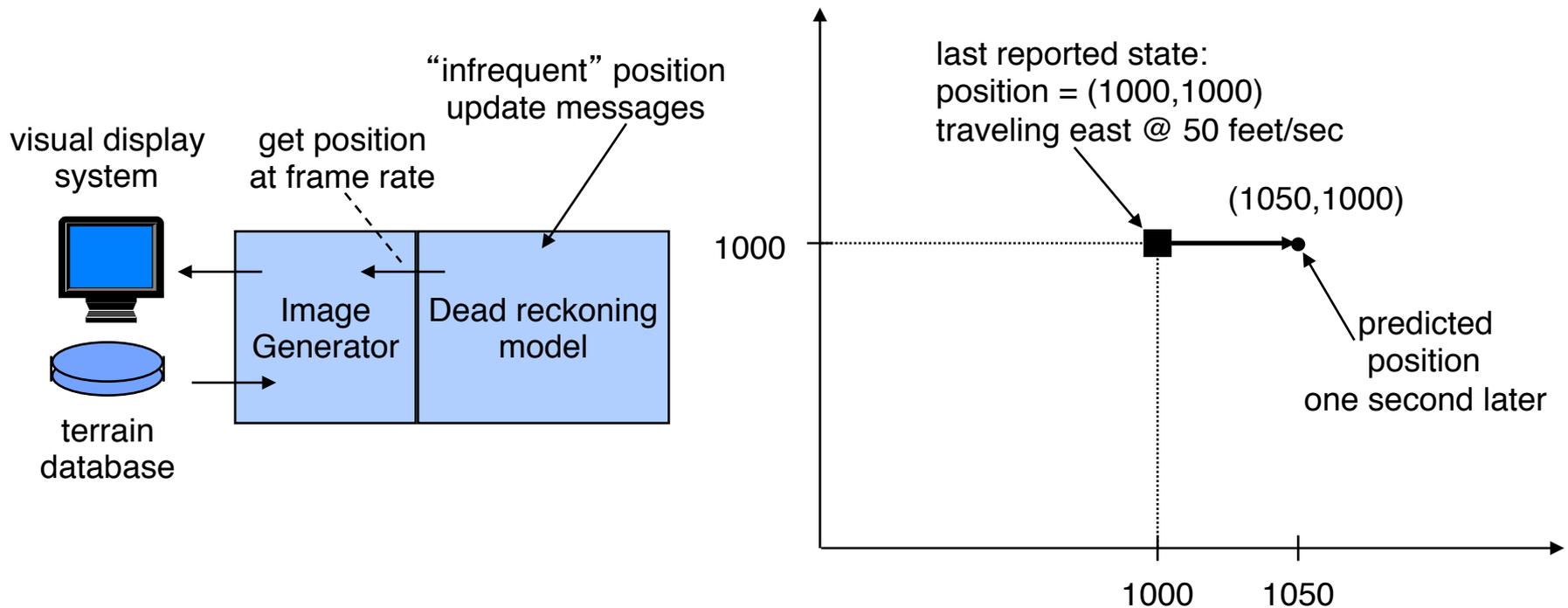


## Limitations

- Position information corresponds to location when the message was sent; doesn't take into account delays in sending message over the network
- Requires generating many messages if there are many vehicles

# Dead Reckoning

- Send position update messages less frequently
- local dead reckoning model predicts the position of remote entities between updates



- When are updates sent?
- How does the DRM predict vehicle position?