Dead Reckoning

Richard M. Fujimoto
Professor

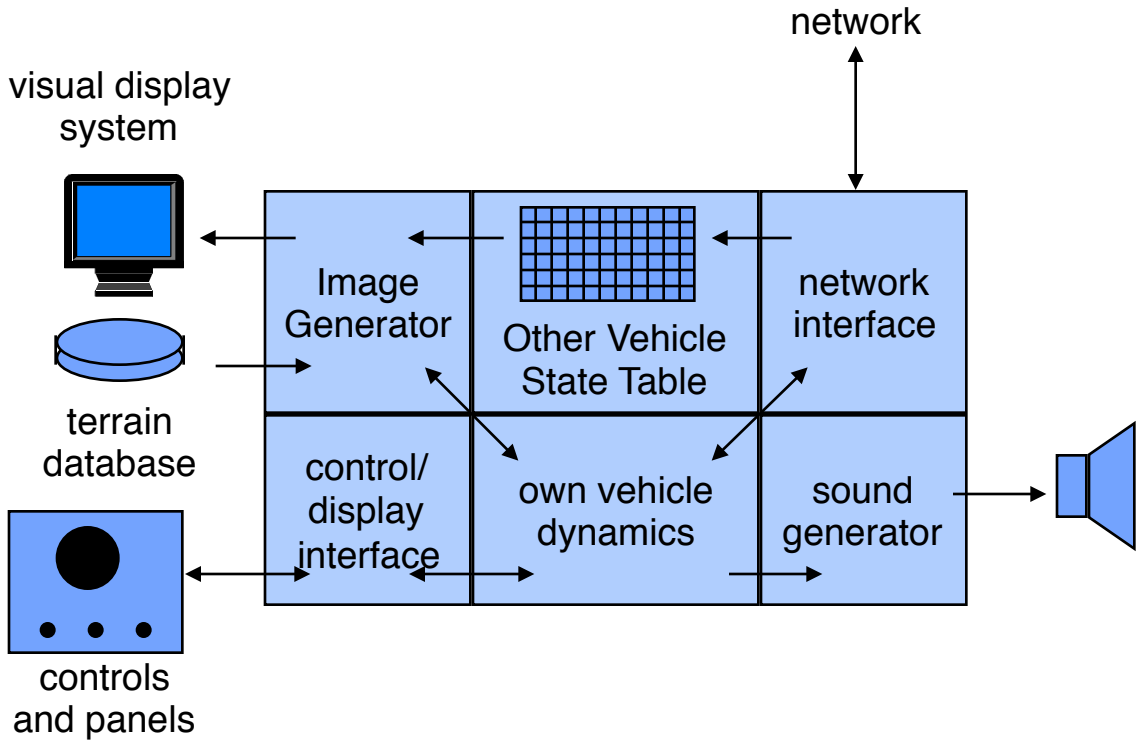
Computational Science and Engineering Division
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0765, USA

http://www.cc.gatech.edu/~fujimoto/

Outline

- Basic Dead Reckoning Model (DRM)
  - Generating state updates
  - Position extrapolation
- Refinements
  - Time compensation
  - Smoothing

A Typical DVE Node Simulator



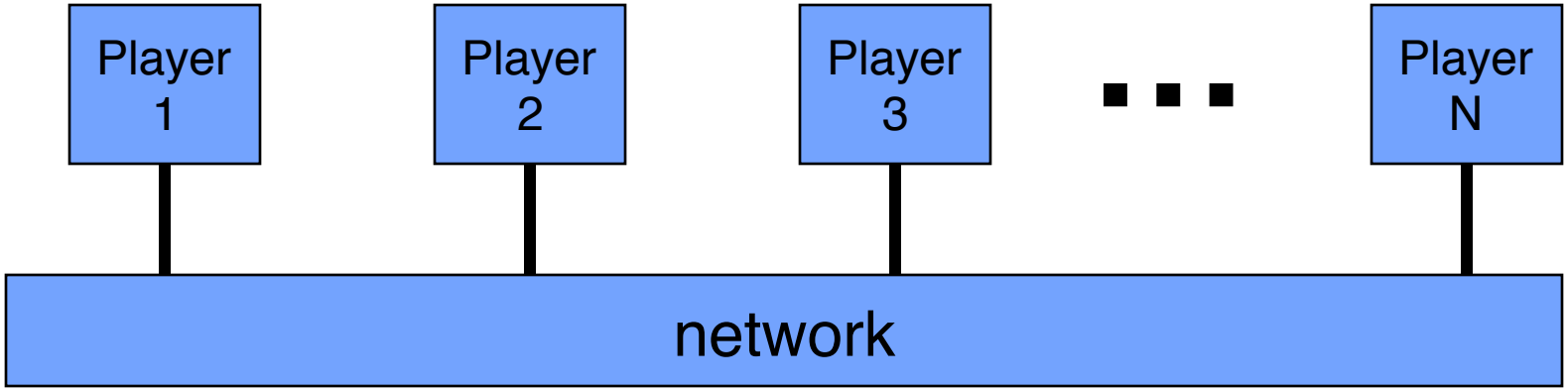
Execute every 1/30th of a second:
- receive incoming messages & user inputs, update state of remote vehicles
- update local display
- for each local vehicle
  - compute (integrate) new state over current time period
  - send messages (e.g., broadcast) indicating new state

Reproduced from Miller, Thorpe (1995), "SIMNET: The Advent of Simulator Networking," Proceedings of the IEEE, 83(8): 1114-1123.

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 1/30th 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

Communication Requirements



- Each player has 1.0 Mbits/sec connection
- DIS: PDU contains 144 bytes (1152 bits)
- Each vehicle generates position update every 1/30 second
  - 34,560 bits per second
- Upper bound: support 29 entities
- Above is extremely optimistic
  - Cannot utilize all of the available bandwidth
  - Other PDUs (e.g., weapon fires) must be generated
  - Multiple entities per human player (synthetic forces)
  - CPU time to process messages often is limiting factor, not bandwidth

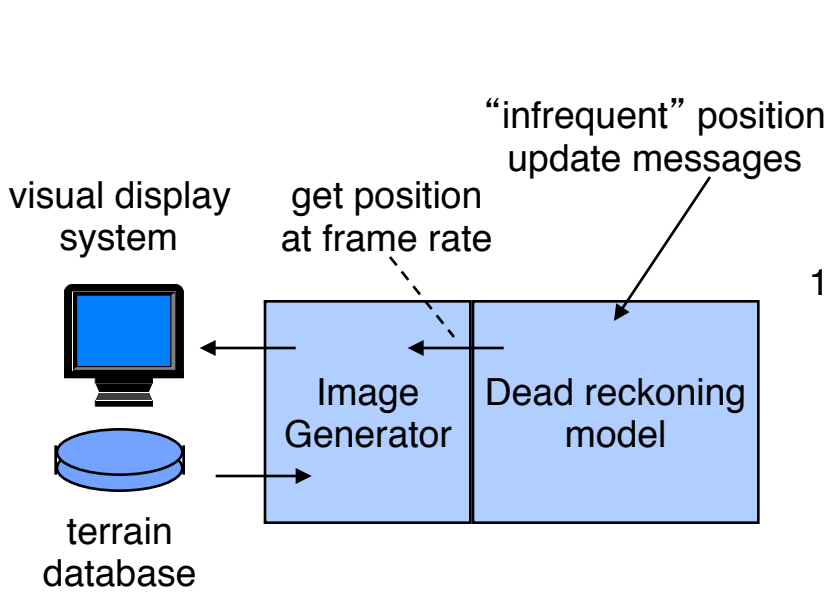
Issues

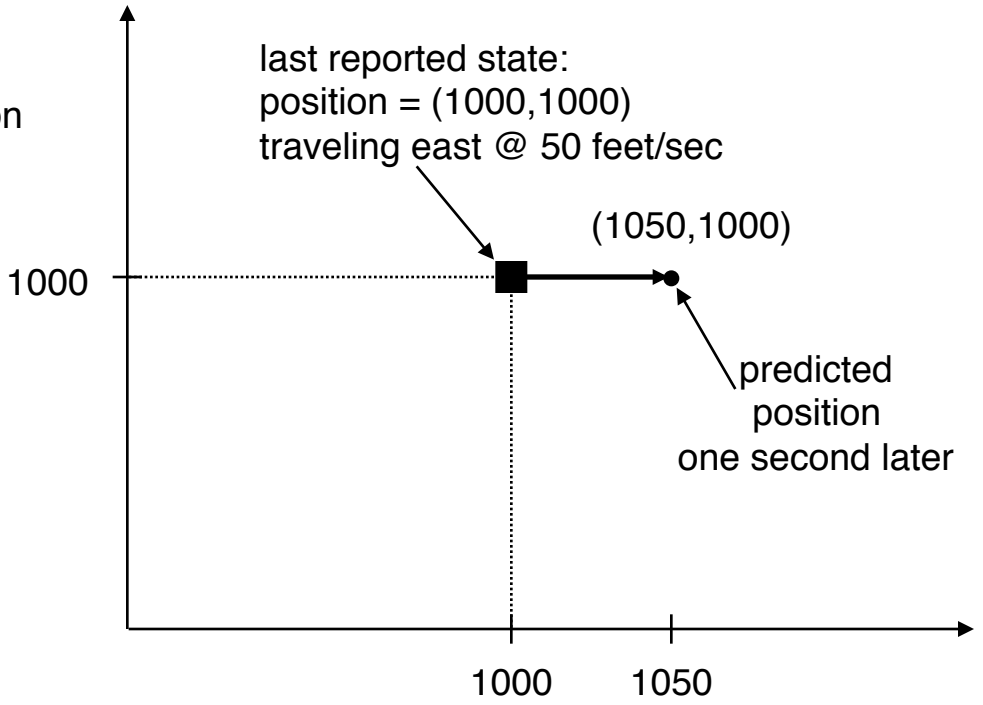
- Requires generating many messages if there are many vehicles; we need ways to economize on communication bandwidth
- Position information corresponds to location when the message was sent; doesn't take into account delays in sending message over the network

Dead reckoning is one technique that attempts to address each of these issues.

Dead Reckoning

- Send position update messages less frequently
- local dead reckoning model predicts the position of remote entities between updates
- Origins: predicting ship locations



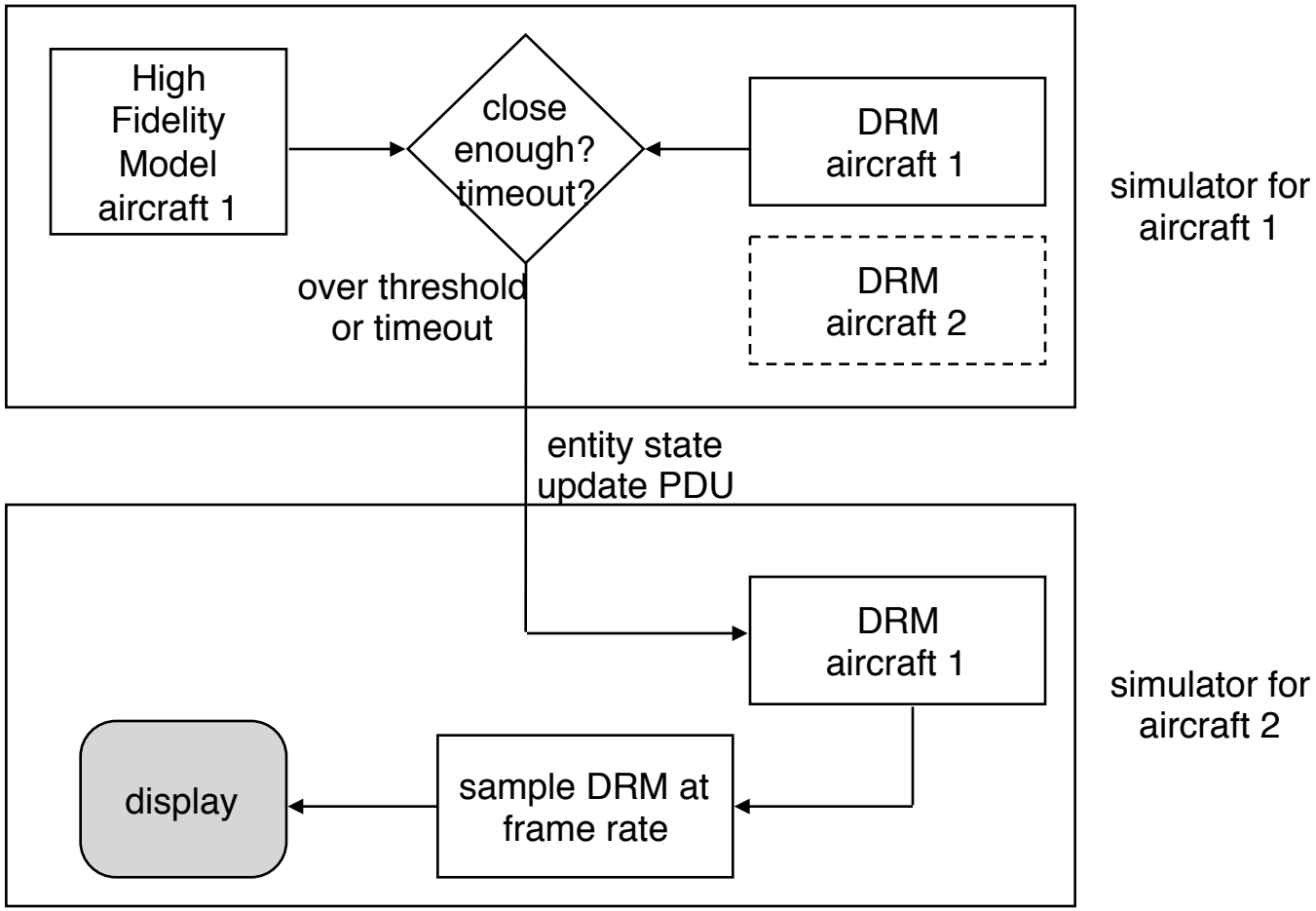


- When are updates sent?
- How does the DRM predict vehicle position?

Re-synchronizing the DRM

When are position update messages generated?

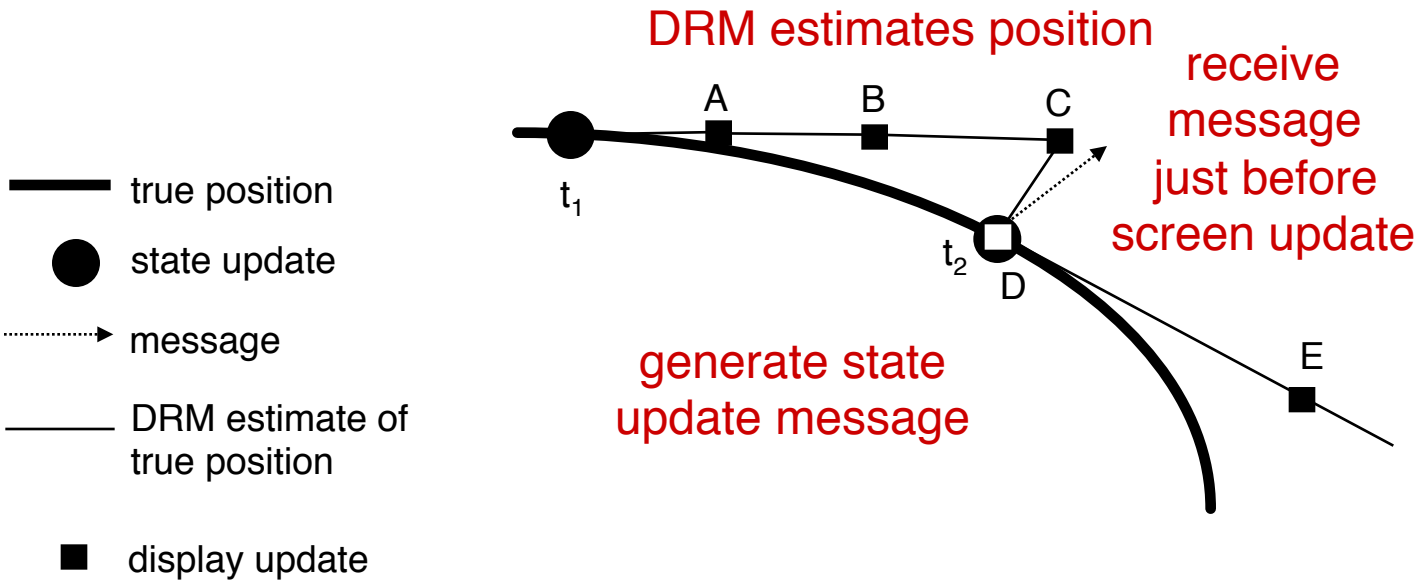
- Compare DRM position with exact position, and generate an update message if error is too large
- Generate updates at some minimum rate, e.g., 5 seconds (heart beats)



Dead Reckoning Models

- P(t) = precise position of entity at time t
- Position update messages: $P(t_1)$, $P(t_2)$, $P(t_3)$ …
- $v(t_i)$, $a(t_i)$ = ith velocity, acceleration update
- DRM: estimate D(t), position at time t
  - t_i = time of last update preceding t
  - $\Delta t = t - t_i$
- 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



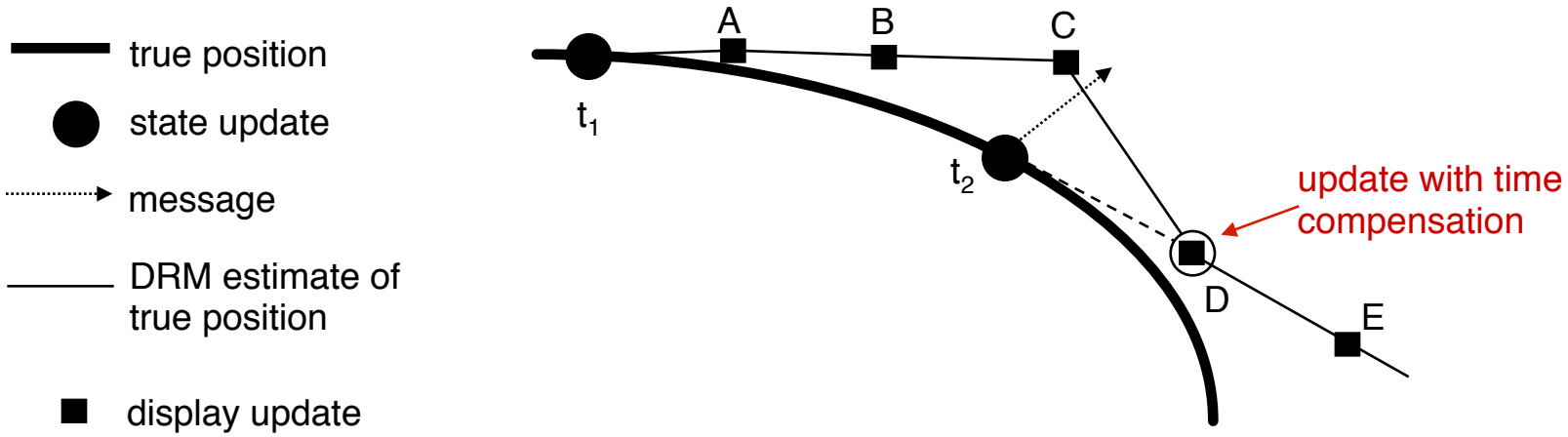
Potential problems:

- 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

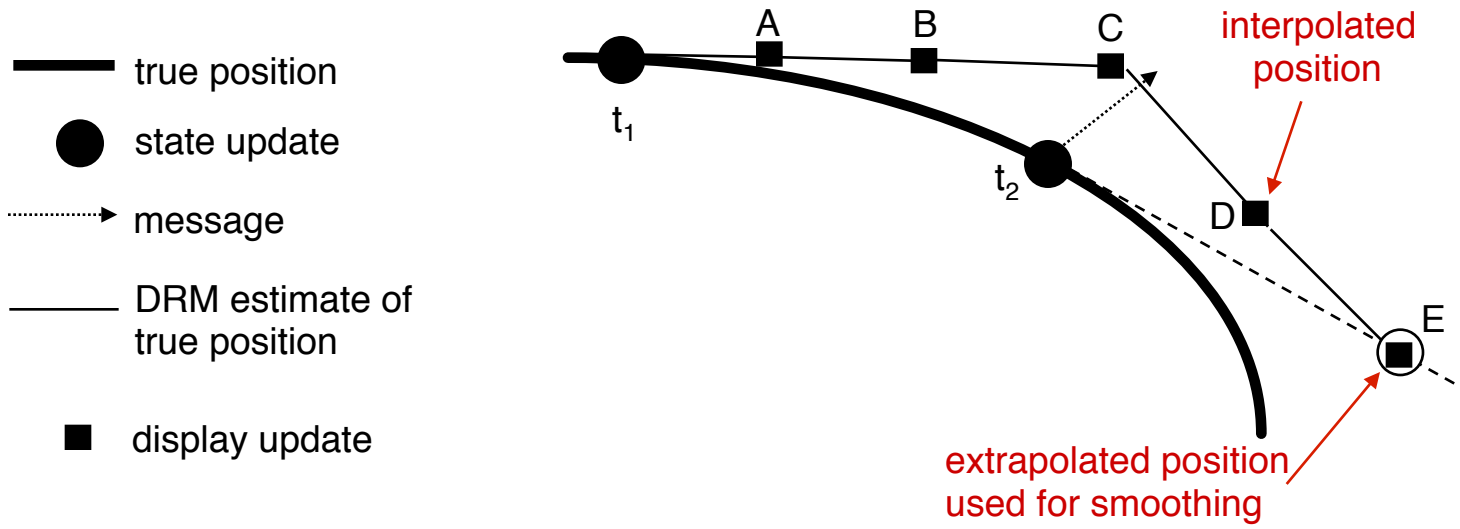
- Add time stamp to message when update is generated (sender time stamp)
- Dead reckon based on message time stamp



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

Summary

- Managing communications is a major issue in implementing distributed simulations
- Dead reckoning model (DRM)
  - Extrapolate current position based on past updates
  - Send update messages when DRM error becoming too large
  - Reduces interprocessor communication
- DRM based on equations of motion
- Time compensation to account for message latency
- Smoothing to avoid “jumps” in display