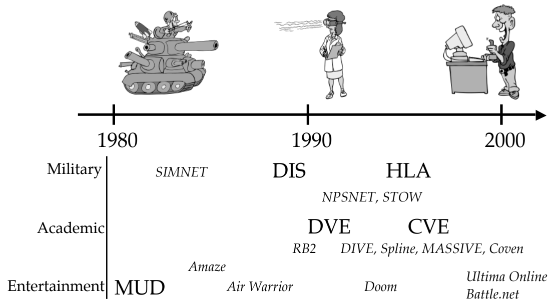
§8.3 Networked Application

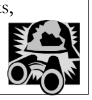
- Department of Defense (DoD)
  - SIMNET
  - Distributed Interactive Simulation (DIS)
  - High-Level Architecture (HLA)
- Academic NVEs
  - PARADISE
  - DIVE
  - BrickNet
  - other academic projects
- Networked games and demos
  - SGI *Flight, Dogfight* and *Falcon A.T.*
  - *Doom*
  - other multiplayer games

History and Evolution



U.S. Department of Defense (DoD)

- The largest developer of networked virtual environments (NVEs) for use as simulation systems
  - one of the first to develop NVEs with its SIMNET system
  - the first to do work on large-scale NVEs
- SIMNET (simulator networking)
  - begun 1983, delivered 1990
  - a distributed military virtual environment developed for DARPA (Defense Advanced Research Projects Agency)
  - develop a 'low-cost' NVE for training small units (tanks, helicopters,…) to fight as a team



SIMNET

- Technical challenges
  - how to fabricate high-quality, low-cost simulators
  - how to network them together to create a consistent battlefield
- Testbed
  - 11 sites with 50–100 simulators at each site
  - a simulator is the portal to the synthetic environment
  - participants can interact/play with others
  - play was unscripted free play
  - confined to the chain of command

SIMNET NSA

Basic components

i. An object-event architecture
ii. A notion of autonomous simulator nodes
iii. An embedded set of predictive modelling algorithms (i.e., 'dead reckoning')

i. Object-Event Architecture

- Models the world as a collection of *objects*
  - vehicles and weapon systems that can interact
  - a single object is usually managed by a single host
  - 'selective functional fidelity'

- Models interactions between objects as a collection of *events*
  - messages indicating a change in the world or object state
- The basic terrain and structures are separate from the collection of objects
  - if the structure can be destroyed then it has to be reclassified as an object, whose state is continually transmitted onto the network

ii. Autonomous Simulator Nodes

- Individual players, vehicles, and weapon systems on the network are *responsible* for transmitting *accurately* their current state
- Autonomous nodes do not interact with the recipients by any other way
- Recipients are responsible for
  - receiving state change information
  - making appropriate changes to their local model of the world
- Lack of a central server
  - single point failures do not crash the whole simulation
  - players can join and leave at any time (persistency)
- Each node is responsible for one or more objects
  - the node has to send update packets to the network whenever its objects have changed enough to notify the other nodes of the change
  - a 'heartbeat' message, usually every 5 seconds

iii. Predictive Modelling Algorithms

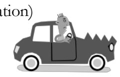
- An embedded and well-defined set of predictive modelling algorithms called *dead reckoning*
- Average SIMNET packet rates:
  - 1 per second for slow-moving ground vehicles
  - 3 per second for air vehicles
- Other packets
  - fire: a weapon has been launced
  - indirect fire: a ballistic weapon has been launced
  - collision: a vehicle hits an object
  - impact: a weapon hits an object

Distributed Interactive Simulation (DIS)

- Derived from SIMNET
  - object-event architecture
  - autonomous distributed simulation nodes
  - predictive modelling algorithms
- Covers more simulation requirements
  - to allow any type of player, on any type of machine
  - to achieve larger simulations
- First version of the IEEE standard for DIS appeared 1993
- Protocol data unit (PDU)
  - determine when each vehicle (node) should issue a PDU
  - the DIS standard defines 27 different PDUs
  - only 4 of them interact with the environment
    - entity state, fire, detonation, and collision
  - the rest of the defined PDUs
    - simulation control, electronic emanations, and supporting actions
    - not supported and disregarded by most DIS applications

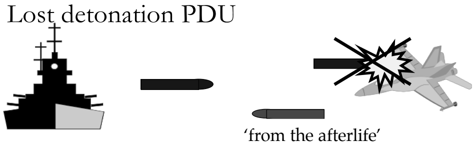
Issuing PDUs

- The vehicle's node is responsible of issuing PDUs
  - entity state PDU
    - when position, orientation, velocity changes sufficiently (i.e., others cannot accurately predict the position any more)
    - as a heartbeat if the time threshold (5 seconds) is reached after the last entity state PDU
  - fire PDU
  - detonation PDU
    - a fired projectile explodes
    - node's vehicle has died (death self-determination)
  - collision PDU
    - vehicle has collided with something
    - detection is left up to the individual node



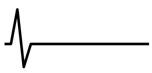
Lost PDUs 1 (2)

- Packets are sent via unreliable UDP broadcast
- State tables may differ among the hosts
- Lost detonation PDU



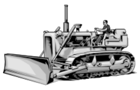
Lost PDUs 2 (2)

- Lost entity state PDU
  - not a big problem
  - larger jumps on the display
- Lost fire PDU
  - receive entity state PDU for which no ghost entry exists
- Lost collision PDU
  - continue to display a vehicle as live
  - next heartbeat packet solves the situation



The Fully Distributed, Heterogeneous Nature of DIS

- Any computer that reads/writes PDUs and manages the state of those PDUs can participate a DIS environment
- The virtual environment can include
  - virtual players (humans at computer consoles)
  - constructive players (computer-driven players)
  - live players (actual weapon systems)
- Problem of the advantages of the low-end machines
  - the less details in the scenery, the better visuality
- Problems with modelling
  - dynamic terrain
    - soil movement
  - environmental effects
    - weather, smoke, dust,…



High-Level Architecture (HLA)

- Aims at providing a general architecture and services for distributed data exchange.
- While the DIS protocol is closely linked with the properties of *military* units and vehicles, HLA does not prescribe any specific implementation or technology.
  - could be used also with non-military applications (e.g., computer games)
  - targeted towards new simulation developments
- HLA was issued as IEEE Standard 1516 in 2000.

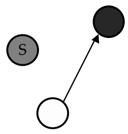
Academic Research

- DoD’s projects
  - large-scale NVEs
  - most of the research is unavailable
  - lack-of-availability, lack-of-generality
- Academic community has reinvented, extended, and documented what DoD has done
  - PARADISE
  - DIVE
  - BrickNet
  - and many more…

PARADISE

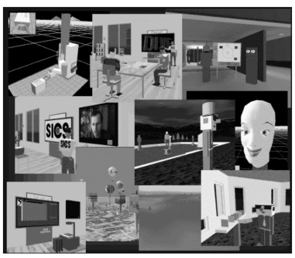
- Performance Architecture for Advanced Distributed Interactive Simulations Environments (PARADISE)
- Initiated in 1993 at Stanford University
- A design for a network architecture for thousands of users

- Assign a different multicast address to each active object
- Object updates similar to SIMNET and DIS
- A hierarchy of area-of-interest servers
  - monitor the positions of objects
  - which multicast addresses are relevant



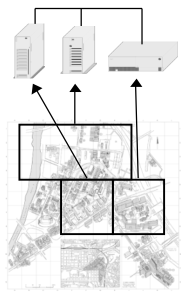
DIVE

- Distributed Interactive Virtual Environment (DIVE)
- Swedish Institute of Computer Science
- To solve problems of collaboration and interaction
- Simulate a large shared memory over a network
- Distributed, fully replicated database
- Entire database is dynamic
  - add new objects
  - modify the existing databases
  - reliability and consistency



BrickNet

- National University of Singapore, started in 1991
- Support for graphical, behavioural, and network modelling of virtual worlds
- Allows objects to be shared by multiple virtual worlds
- No replicated database
- The virtual world is partitioned among the various clients



Other Academic Projects

- MASSIVE
  - different interaction media: graphics, audio and text
  - awareness-based filtering: each entity expresses a focus and nimbus for each medium
- Distributed Worlds Transfer and Communication Protocol (DWTP)
  - each object can specify whether a particular event requires a reliable distribution and what is the event's maximum update frequency
- Real-Time Transport Protocol (RTP/I)
  - ensures that all application instances look as if all operations have been executed in the same order
- Synchronous Collaboration Transport Protocol (SCTP)
  - collaboration on closely coupled, highly synchronized tasks
  - the interaction stream has critical messages (especially the last one) which are sent reliably, while the rest are sent by best effort transport

Networked Demos and Games

- SGI *Flight*
  - 3D aeroplane simulator demo for Silicon Graphics workstation, 1983–84
    - serial cable between two workstations
    - Ethernet network
    - users could see each other's planes, but no interaction

- SGI *Dogfight*
  - modification of *Flight*, 1985
  - interaction by shooting
  - packets were transmitted at frame rate → clogged the network
  - limited up to ten players
- *Falcon A.T.*
  - commercial game by Spectrum Holobyte, 1988
  - dogfighting between two players using a modem

Networked Games: *Doom*

- id Software, 1993
- First-person shooter (FPS) for PCs
- Part of the game was released as shareware in 1993
  - extremely popular
  - created a gamut of variants
- Flooded LANs with packets at frame rate

Networked Games: 'First Generation'

- Peer-to-peer architectures
  - each participating computer is an equal to every other
  - inputs and outputs are synchronized
  - each computer executes the same code on the same set of data
- Advantages:
  - determinism ensures that each player has the same virtual environment
  - relatively simple to implement
- Problems:
  - persistency: players cannot join and leave the game at will
  - scalability: network traffic explodes with more players
  - reliability: coping with communication failures
  - security: too easy to cheat

Networked Games: 'Second Generation'

- Client-server architectures
  - one computer (a server) keeps the game state and makes decisions on updates
  - clients convey players' input and display the appropriate output but do not inlude (much) game logic
- Advantages:
  - generates less network traffic
  - supports more players
  - allows persistent virtual worlds
- Problems:
  - responsiveness: what if the connection to the server is slow or the server gets overburdened?
  - security: server authority abuse, client authority abuse

Networked Games: 'Third Generation'

- Client-server architecture with prediction algorithms
  - clients use dead reckoning
- Advantages:
  - reduces the network traffic further
  - copes with higher latencies and packet delivery failures
- Problems:
  - consistency: if there is no unequivocal game state, how to solve conflicts as they arise?
  - security: packet interception, look-ahead cheating

Networked Games: 'Fourth Generation'

- Generalized client-server architecture
  - the game state is stored in a server
  - clients maintain a subset of the game state locally to reduce communication
- Advantages:
  - traffic between the server and the clients is reduced
  - clients can response more promptly
- Problems:
  - boundaries: what data is kept locally in the client?
  - updating: does the subset of game state change over time?
  - consistency: how to solve conflicts as they occur?

Communication Layers (Revisited)

- physical platform
  - bandwidth, latency
  - unicasting, multicasting, broadcasting
  - TCP/IP, UDP/IP
- logical platform
  - peer-to-peer, client-server, server-network
  - centralized, replicated, distributed
- networked application
  - military simulations, networked virtual environments
  - multiplayer computer games