Dynamic Server Allocation in a Real-Life Deployable Communications Architecture for Networked Games

Peter Quax
Bart Cornelissen
Jeroen Dierckx
Gert Vansichem
Wim Lamotte

Hasselt University & Androme NV / Belgium
• Request by Flemish public broadcasting company (VRT)
  – 3D virtual environments for story-telling
  – Goal:
    • Support for existing TV programs (characters, story lines,...)
    • Have people ‘create’ new content for possible future programs
  – Problems
    • Financial issues (hosting)
    • Bad experience with previous experiment
    • Success rate for programs is unknown
Identified issues

- **Scalability**
  - Single (massive) world is required for each story – don’t use shards or instances
  - Keep initial investment costs low, but make it easy to add capacity (unlike Second Life)

- **Manageability**
  - User-generated content and user actions need to be kept under control (children)
  - Designate trusted sources and parties
  - Use a client/server architecture; peer-to-peer is superior for scalability but management issues remain problematic.

- **Existing solutions**
  - Sun Game Server technology, MultiVerse, *Eve-online*
The ALVIC architecture

- Architecture for Large-scale Virtual Interactive Communities
  - As presented in NetGames 2003 at EA
  - Peer-to-peer system based on multicast communication
    - Spatial subdivision scheme coupled to multicast addresses
    - Clients were able to control downstream bandwidth by changing the size & shape of the area-of-interest
  - Problems
    - Required clients to be able to send multicast traffic to the WAN (or tunneling)
    - Very hard to manage from content provider point of view
Introducing ALVIC-NG

• Second generation framework
  – No longer peer-to-peer
    • Too many practical issues (deployment, consistency,...)
    • Content providers want control over the system (moderation,...)

• Should tackle the following issues
  – Efficient spatial subdivision scheme
  – Highly dynamic resource allocation at server-side
  – Minimize configuration and overhead needed at client-side
ALVIC-NG architecture

- What’s specific?
  - Additional layer between the servers and the clients: *proxies*
  - *Region management system* (RMS) that links spatial subdivision scheme to server allocation

- Resource allocation is dynamic
  - No single central database that maintains ‘state’
  - Server infrastructure can grow dynamically depending on the # of users/subscribers -> lower initial investment
ALVIC-NG elements

- Proxy servers
  - Tunneling of traffic that is normally sent between clients and ‘world’ servers
    - Significantly reduces the # of connections for both clients and servers
    - Forwarding and packet inspection
  - Caching of data
    - Mostly non-state-related information
  - Specific proxies can be selected by client with regards to several parameters
    - Location (minimal RTT values)
    - Load (processing, network,...)
  - Pool of proxies is managed by central (trusted) entity
**ALVIC-NG elements**

- **Logic servers**
  - Manage parts of the virtual world
    - Keep track of object state
    - Manage NPCs by executing scripts
    - Distinguish between ‘levels’ of persistency
  - Assignment of logic servers to regions (spatial subdivision scheme) is highly dynamic
    - The system can manage if areas become overcrowded
    - References are maintained by the Region Management System (RMS)
ALVIC-NG elements

• Authentication
  – Handled by external providers
    • E.g. Electronic Identity Card
  – All servers are known to the authentication system
    • Eliminates many of the chances to introduce ‘rogue’ proxy/logic servers

• Asset database
  – Various types of elements
    • Meshes
    • Scripts for animated objects
    • Behaviors for NPCs
  – Assets are downloaded to Logic servers as needed
ALVIC-NG elements

- Region Management System
  - Maintains a mapping between regions and logic servers (like DNS)
  - RMS tracks several parameters
    - Load
      - Processing
      - Network usage
      - # of active clients in region
    - Exchange of information through SNMP-like protocol
      - RMS not only queries, but actively tries to resolve problems
    - Logic server failure -> assignment of region to other server(s)
    - Overcrowding -> send instructions to logic servers to split regions and update mapping tables
• Step 1
  – Client authenticates using his/her credentials
    • In our case: Electronic Identity Card
  – (ordered) list of available proxy servers is retrieved
    • List is also maintained by the RMS
• Step 2
  – Choose the proxy server to connect to according to a metric
    • Approximate determination of delay between client and proxy based on e.g. WHOIS records
    • Current load on proxy (network, processing)
Step 2

- Connections to proxy are established
  - TCP connection for control
  - UDP channel for ‘bulk’ data
- Authentication channel may be left open if needed
  - Changing keys during session
• Step 3 & 4
  – Client announces his/her position in the virtual world to the proxy
    • Client does not know about spatial subdivision scheme!
    • Proxy queries the RMS to know what Logic server is responsible for the region
• Step 5 & 6
  – Proxy connects to Logic server that handles the part of the world the client is located in
    • State of other objects is retrieved and forwarded to the client
  – Additional connections are made as the area-of-interest changes
    • Connections no longer needed are dropped
    • # of open connections between proxies and logic servers can be optimized
• Step 7
  - Logic server knows about the relative importance of state information
    • Some state requires frequent storage on persistent media (hard drive) -> e.g. financial transactions
    • Operations on state are handled in memory
  - Use of off-the-shelf (R)DBMS systems
    • Either 1 database system per logic server or multiple servers per database
• Step 8
  – Proxy does packet inspection of ‘state’ packets sent by client
  – In case of region boundary crossing:
    • Establish connection(s) to the new logic server – if not already connected
    • Remove existing connections if no other clients require updates from the ‘old’ region
  – Spatial subdivision scheme needs to support fast boundary determination
• Step 9 and 10
  – Direct connections between logic servers are needed to:
    • Exchange state of single client at boundary crossing
    • Exchange ‘bulk’ state information when a region is split/merged or a new logic server is assigned
  – In case of logic server failure:
    • Retrieve information from the (R)DBMS system that provided persistent storage and restore state
Spatial subdivision

- Region splitting/merging is decided upon by the RMS
  - RMS has an (continually updated) global overview of the load distribution over the logic servers
  - Decisions are based on freely determined metrics, but in most cases:
    - # of clients in region
    - Processing load vs capacity
    - Bandwidth usage vs capacity

- The system does not go down when the topology changes!
  - However, a disruption in the experience is unavoidable
  - Major improvement over than the classic system (e.g. Second Life) that can not cope with overcrowding
Scalability testing

• Determine the overhead introduced by additional components (proxies)
  – Each proxy server has to support a large # of users
    • Economic impact (additional cost)
    • Reduced # of connections for logic servers
  – Without proxies, the system resembles traditional approaches

• How to determine overhead
  1. Modeling the bandwidth usage/processing requirements
  2. Test setups that come close to real life — simulation

• Advantages of simulation
  – Make sure that the implementation works
  – Modeling can overlook certain issues
Scalability testing

- Use the actual client software
  - But strip the 3D visualization
  - Control all instances through a central process (Bot Server)
  - Individual clients behave under control of LUA scripts -> randomness

- Execution
  - Run large number of concurrent processes on a dedicated cluster
  - Use the actual implementations of the various servers
Scalability testing

- Visual check of simulation
  - Single client application that is controlled by human operator
  - Can provide overview of spatial subdivision scheme and client distribution
Scalability testing

• Load on logic servers is not examined
  – Is heavily dependent on the type of application
    • Player-player interactions vs. NPC behaviors
  – There is existing work that can be referred to
    • Results will be at least as good as in existing work, as fewer connections need to be managed

• Simulation parameters
  – Use a state update rate of 3 per second
    • A ‘smoothing’ algorithm is often used/required (e.g. dead reckoning)
    • Value is representative for real-life applications
  – RTT is the “load” metric, not raw CPU usage
    • RTT measured between the “send” action of the client and the reception of an echo of the update
    • Network delay on the Gbit LAN is negligible – nearly all delay is introduced by software
  – Cut-off value for interactivity = 50 ms
    • Seems rather low, but does not include network-induced delay
Scalability testing

- **Tests runs**
  - Scenarios with between one and five proxies
  - Each run is repeated five times to even out the results
  - Randomness is guaranteed by the scripted behaviors
- **Connections between client instances and proxies are established in round-robin**
  - Ensures an even load on the proxies
- **Test results show absolute figures**
  - In practice, trends are more important
  - Cluster is made up of relatively low-end hardware
Result 1:

# of clients vs # of proxies
Result 1:

# of clients vs # of proxies

- **Parameters**
  - Data points are sampled at 10 second intervals
  - Each bot server spawns at least 1 client per second
  - # of bot servers equals # of proxies

- **Observations**
  - Simulation with 1 proxy gets overloaded around 625 clients
Result 2: acceptable # of clients vs # of proxies
Acceptable # of clients given # of proxies

- **Parameters**
  - Interactivity threshold of 50 ms

- **Observations**
  - Scales nearly linear between scenarios
  - Good indication that scalability is ensured
Extended test
Future Work

• Next steps
  – Re-run tests on a 100+ node cluster
  – ALVIC-NG also includes a conferencing system
    • Based on similar spatial subdivision scheme
    • Can efficiently distribute audio/video streams between large # of participants