Massive Multiplayer Online First Person Shooter as Peer–to–Peer game

Christian Grümme
gruemme@mi.fu-berlin.de

Freie Universität Berlin

July 7, 2008

Objective and Approach

- Objective
- Approach
- Synchronizing

Implementation

- The Sectorizier
- nodes
- Game adjustment

Evaluation

- Setting
- Results
- Scalability

Conclusion

- Conclusion
- Future research

Objective – Server-Client architecture

- Centralized:
  - No consistent problems
  - Less cheating capabilities
  - Single point of failure
- Clients share the server's bandwidth
- The server becomes the bottleneck of this architecture, especially with a growing amount of clients.

Objective – Server-Client architecture

- Centralized:
  - No consistent problems
  - Less cheating capabilities
  - Single point of failure
- Clients share the server's bandwidth
- The server becomes the bottleneck of this architecture, especially with a growing amount of clients.

Objective – Server-Client architecture

- Centralized:
  - No consistent problems
  - Less cheating capabilities
  - Single point of failure
- Clients share the server's bandwidth
- The server becomes the bottleneck of this architecture, especially with a growing amount of clients.

Objective – Server-Client architecture

- Centralized:
  - No consistent problems
  - Less cheating capabilities
  - Single point of failure
- Clients share the server's bandwidth
- The server becomes the bottleneck of this architecture, especially with a growing amount of clients.

Objective – Server-Client architecture

- Centralized:
  - No consistent problems
  - Less cheating capabilities
  - Single point of failure
- Clients share the server's bandwidth
- The server becomes the bottleneck of this architecture, especially with a growing amount of clients.

Objective – Server-Client architecture

- Centralized:
    - No consistent problems
    - Less cheating capabilities
    - Single point of failure
- Clients share the server's bandwidth
- The server becomes the bottleneck of this architecture, especially with a growing amount of clients.

Objective – Peer–to–Peer architecture

- Decentralized
  - Robustness to the failure of single nodes
  - Synchronization is needed to preserve consistence of the game world
- Bandwidth with a growing amount of participants

Objective – Peer–to–Peer architecture

- Decentralized
  - Robustness to the failure of single nodes
  - Synchronization is needed to preserve consistence of the game world
- Bandwidth with a growing amount of participants

Objective – Peer–to–Peer architecture

- Decentralized
  - Robustness to the failure of single nodes
  - Synchronization is needed to preserve consistence of the game world
- Bandwidth with a growing amount of participants

Objective – Peer–to–Peer architecture

- Decentralized
  - Robustness to the failure of single nodes
  - Synchronization is needed to preserve consistence of the game world
- Bandwidth with a growing amount of participants

Objective

- Replacing the disadvantages of the Server-Client architecture by the advantages of the Peer–to–Peer architecture
- Solve disadvantages of the Peer–to–Peer architecture
- Constraint: A generic approach in order to change a minimum on the game logic itself -> So this approach can be applied to a large number of applications

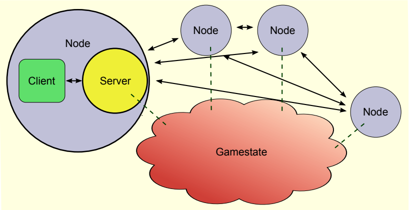
Objective

- Replacing the disadvantages of the Server-Client architecture by the advantages of the Peer–to–Peer architecture
- Solve disadvantages of the Peer–to–Peer architecture
- Constraint: A generic approach in order to change a minimum on the game logic itself -> So this approach can be applied to a large number of applications

Objective

- Replacing the disadvantages of the Server-Client architecture by the advantages of the Peer–to–Peer architecture
- Solve disadvantages of the Peer–to–Peer architecture
- Constraint: A generic approach in order to change a minimum on the game logic itself -> So this approach can be applied to a large number of applications

Approach 1 – Forming a server node



One modified server and an unmodified client are forming a server node

Approach 2 – The tasks of a modified server

A modified server has to fulfill the following tasks:

- Providing a game world for the client
- Realizing "real time" communication to other server nodes
- Communicating just to those server nodes that are relevant

Approach 2 – The tasks of a modified server

A modified server has to fulfill the following tasks:

- Providing a game world for the client
- Realizing “real time“ communication to other server nodes
- Communicating just to those server nodes that are relevant

Approach 2 – The tasks of a modified server

A modified server has to fulfill the following tasks:

- Providing a game world for the client
- Realizing “real time“ communication to other server nodes
- Communicating just to those server nodes that are relevant

Approach 2 – The tasks of a modified server

A modified server has to fulfill the following tasks:

- Providing a game world for the client
- Realizing “real time“ communication to other server nodes
- Communicating just to those server nodes that are relevant

Zoning 1 – Conditions

- A sectorization partitions the game world into sectors
- Each sector has to be controlled by at least one server node
- A sector of interests for a server node is a sector in that its clients has its location in or that is neighboring such a sector

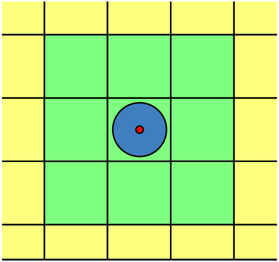
Zoning 1 – Conditions

- A sectorization partitions the game world into sectors
- Each sector has to be controlled by at least one server node
- A sector of interests for a server node is a sector in that its clients has its location in or that is neighboring such a sector

Zoning 1 – Conditions

- A sectorization partitions the game world into sectors
- Each sector has to be controlled by at least one server node
- A sector of interests for a server node is a sector in that its clients has its location in or that is neighboring such a sector

Zoning 2 – Example 1



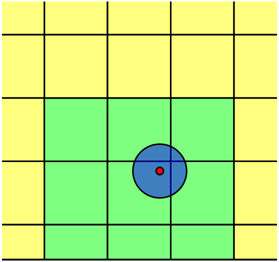
🟥 Client's location

🟦 Client's sight of view

🟨 Sector with no interest

🟩 Sector of interest

Zoning 3 – Example 2



- 🟥 Client's location
- 🟦 Client's sight of view
- 🟨 Sector with no interest
- 🟩 Sector of interest

Request 1 – Postulations

- Each server node has to control its sector of interests
- Each server node has a complete list of the other server nodes with an unique ID (SNID) and an URI
- Each server node has a sector table which maps each sector to a server node that is in control of it (current or last seen, so may be outdated)

Request 1 – Postulations

- Each server node has to control its sector of interests
- Each server node has a complete list of the other server nodes with an unique ID (SNID) and an URI
- Each server node has a sector table which maps each sector to a server node that is in control of it (current or last seen, so may be outdated)

Request 1 – Postulations

- Each server node has to control its sector of interests
- Each server node has a complete list of the other server nodes with an unique ID (SNID) and an URI
- Each server node has a sector table which maps each sector to a server node that is in control of it (current or last seen, so may be outdated)

Request 2 – Request Algorithm

Request chain 1

```
1.  Ask last seen SNID controlling s
2.  IF SNID is not controlling s THEN
3.        overwrite entry of s in the sector table
                   with the sector table entry of SNID
4.        GOTO 1
```

Routing

- There is no hardware based multicast available on the Internet
- Degree of a node can be reduced by taking hops to other nodes
- Just useful when the time difference is acceptable or data can be aggregated

Routing

- There is no hardware based multicast available on the Internet
- Degree of a node can be reduced by taking hops to other nodes
- Just useful when the time difference is acceptable or data can be aggregated

Routing

- There is no hardware based multicast available on the Internet
- Degree of a node can be reduced by taking hops to other nodes
- Just useful when the time difference is acceptable or data can be aggregated

Synchronizing

- Synchronizing time between server nodes is a big problem
- Within an ideal environment the Network Time Protocol can just synchronize with an accuracy of 20 ms
- A solution would be a logical time within the game

Synchronizing

- Synchronizing time between server nodes is a big problem
- Within an ideal environment the Network Time Protocol can just synchronize with an accuracy of 20 ms
- A solution would be a logical time within the game

Synchronizing

- Synchronizing time between server nodes is a big problem
- Within an ideal environment the Network Time Protocol can just synchronize with an accuracy of 20 ms
- A solution would be a logical time within the game

Implementation 1 – Parts

The Implementation is divided into three parts:

- The Sectorizier
- The middleware *nodes*
- Other miscellaneous modifications on the original server

Implementation 1 – Parts

The Implementation is divided into three parts:

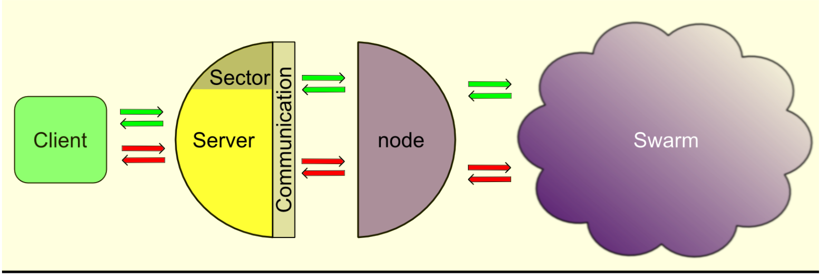
- The Sectorizier
- The middleware *nodes*
- Other miscellaneous modifications on the original server

Implementation 1 – Parts

The Implementation is divided into three parts:

- The Sectorizier
- The middleware *nodes*
- Other miscellaneous modifications on the original server

Implementation 2 – Organization



🟩 Reliable Connection

🟥 Unreliable Connection

The Sectorizier 1 – Function

The Sectorizier . . .

- analyzes a given game map
- generates an XML–file that describes the sectorization of this map
- is implemented in ANSI C
- uses the libxml2 — the XML C parser and toolkit developed for the Gnome project

The Sectorizier 1 – Function

The Sectorizier ...

- analyzes a given game map
- generates an XML–file that describes the sectorization of this map
- is implemented in ANSI C
- uses the libxml2 — the XML C parser and toolkit developed for the Gnome project

The Sectorizier 1 – Function

The Sectorizier . . .

- analyzes a given game map
- generates an XML–file that describes the sectorization of this map
- is implemented in ANSI C
- uses the libxml2 — the XML C parser and toolkit developed for the Gnome project

The Sectorizier 1 – Function

The Sectorizier . . .

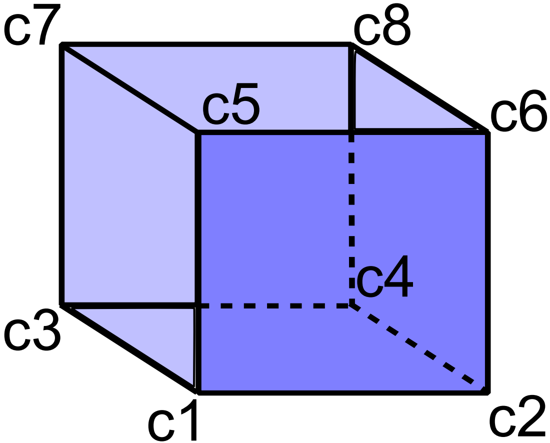
- analyzes a given game map
- generates an XML–file that describes the sectorization of this map
- is implemented in ANSI C
- uses the libxml2 — the XML C parser and toolkit developed for the Gnome project

The Sectorizier 1 – Function

The Sectorizier . . .

- analyzes a given game map
- generates an XML–file that describes the sectorization of this map
- is implemented in ANSI C
- uses the libxml2 — the XML C parser and toolkit developed for the Gnome project

The Sectorizier 2 – Ascending cube



nodes – Function

nodes . . .

- is a JAVA-based middleware
- can be adapt to other applications
- introduces a new abstract data type, the Request Queue
- has a swarm table and a sector table as main data structures
- got a profiler

nodes – Function

nodes . . .

- is a JAVA-based middleware
- can be adapt to other applications
- introduces a new abstract data type, the Request Queue
- has a swarm table and a sector table as main data structures
- got a profiler

nodes – Function

nodes . . .

- is a JAVA-based middleware
- can be adapt to other applications
- introduces a new abstract data type, the Request Queue
- has a swarm table and a sector table as main data structures
- got a profiler

nodes – Function

nodes . . .

- is a JAVA-based middleware
- can be adapt to other applications
- introduces a new abstract data type, the Request Queue
- has a swarm table and a sector table as main data structures
- got a profiler

nodes – Function

nodes . . .

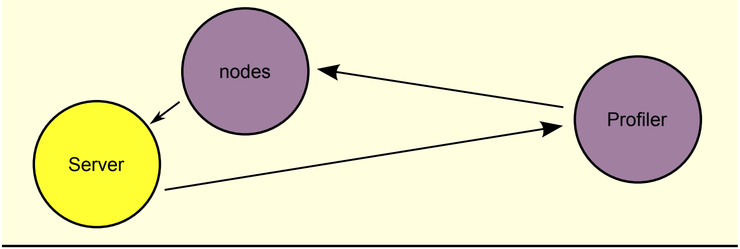
- is a JAVA-based middleware
- can be adapt to other applications
- introduces a new abstract data type, the Request Queue
- has a swarm table and a sector table as main data structures
- got a profiler

nodes – Function

nodes . . .

- is a JAVA-based middleware
- can be adapt to other applications
- introduces a new abstract data type, the Request Queue
- has a swarm table and a sector table as main data structures
- got a profiler

The profiler



■ Unreliable Connection

Game adjustment

Modifications

- The Sectorizier is integrated
- Additional command line parameters
- Communication with *nodes*

Game adjustment

Modifications

- The Sectorizier is integrated
- Additional command line parameters
- Communication with *nodes*

Game adjustment

Modifications

- The Sectorizier is integrated
- Additional command line parameters
- Communication with *nodes*

Small map

Run No.	Nodes	Computers	Profilers
1	10	10	2
2	20	10	4
3	40	10	8

Table: Test run on the small map — q3dm1

Big map

Run No.	Nodes	Computers	Profilers
4	20	10	5
5	20	10	8
6	40	10	20
7	40	10	10
8	80	10	12
9	80	10	10
10	40	4	12
11	20	4	8
12	40	20	10
13	58	29	10

Table: Test run on the big map — sector12_12

Evaluation Graphic – Big map

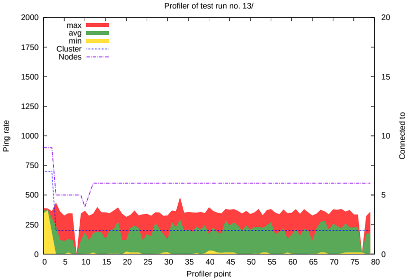


Figure: A profiler of test run 13

Evaluation Graphic – Small map

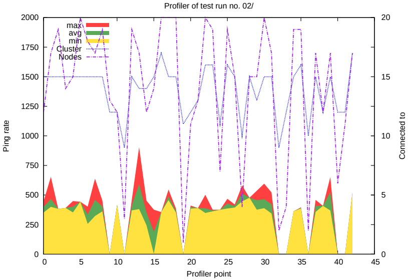


Figure: A profiler of test run 02

Problems

- Game based limitation of the number of players
- Microsoft Windows TCP–Limit (10 TCP–Connection per second)

Problems

- Game based limitation of the number of players
- Microsoft Windows TCP–Limit (10 TCP–Connection per second)

Evaluation Graphic – Problems

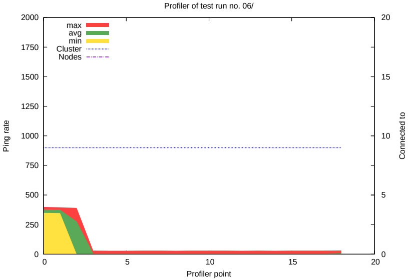


Figure: A profiler of test run 06

Bandwidth consumption

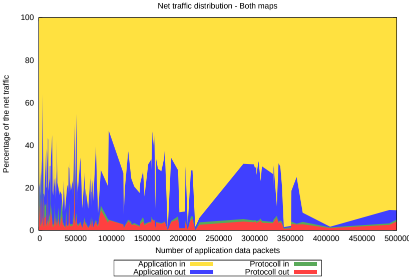


Figure: The visualization of all samples

Conclusion

- Objective was achieved
- But with reservations (TCP–limit)

Conclusion

- Objective was achieved
- But with reservations (TCP–limit)

Future research

- A sectorisation that consider map characteristics
- Cheating protection
- Multi–client server and client migration
- Other applications
- Implementing the presented Routing
- Converting to a hybrid system

Future research

- A sectorisation that consider map characteristics
- Cheating protection
- Multi–client server and client migration
- Other applications
- Implementing the presented Routing
- Converting to a hybrid system

Future research

- A sectorisation that consider map characteristics
- Cheating protection
- Multi–client server and client migration
- Other applications
- Implementing the presented Routing
- Converting to a hybrid system

Future research

- A sectorisation that consider map characteristics
- Cheating protection
- Multi–client server and client migration
- Other applications
- Implementing the presented Routing
- Converting to a hybrid system

Future research

- A sectorisation that consider map characteristics
- Cheating protection
- Multi–client server and client migration
- Other applications
- Implementing the presented Routing
- Converting to a hybrid system

Future research

- A sectorisation that consider map characteristics
- Cheating protection
- Multi–client server and client migration
- Other applications
- Implementing the presented Routing
- Converting to a hybrid system

End

Thank you for listening

`http://page.mi.fu-berlin.de/gruemme/snp2p/`