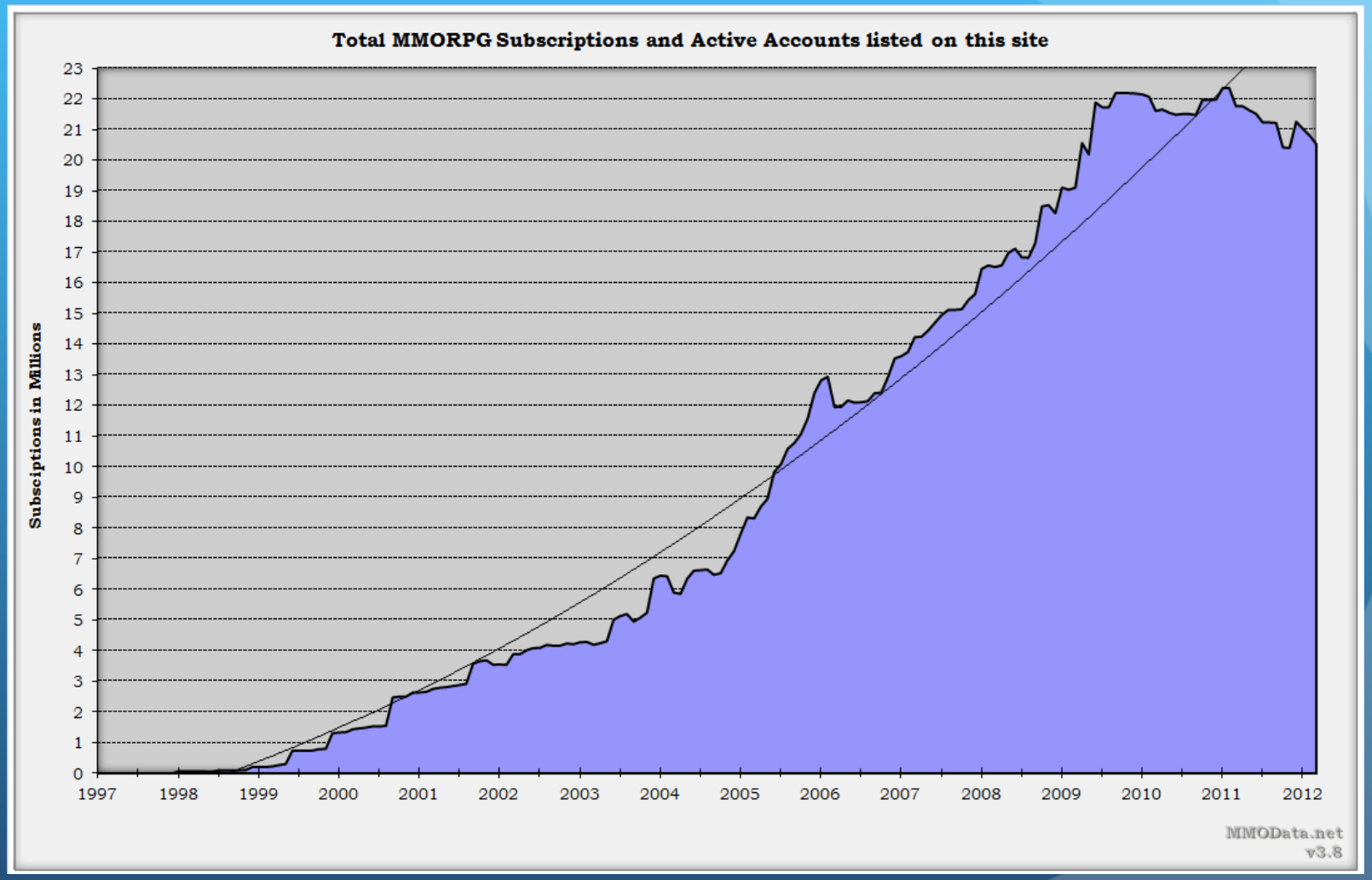
Ch 16 Game System Architecture

Prof. Magda El Zarki
Dept. of CS
UC Irvine, CA
Email: elzarki@uci.edu
http://www.ics.uci.edu/~magda

Overview of MMOG

- Massively MultiUser Online Games: MMOGs
- Classification
- Taxonomy
- Current Systems
- Issues and Future Trends

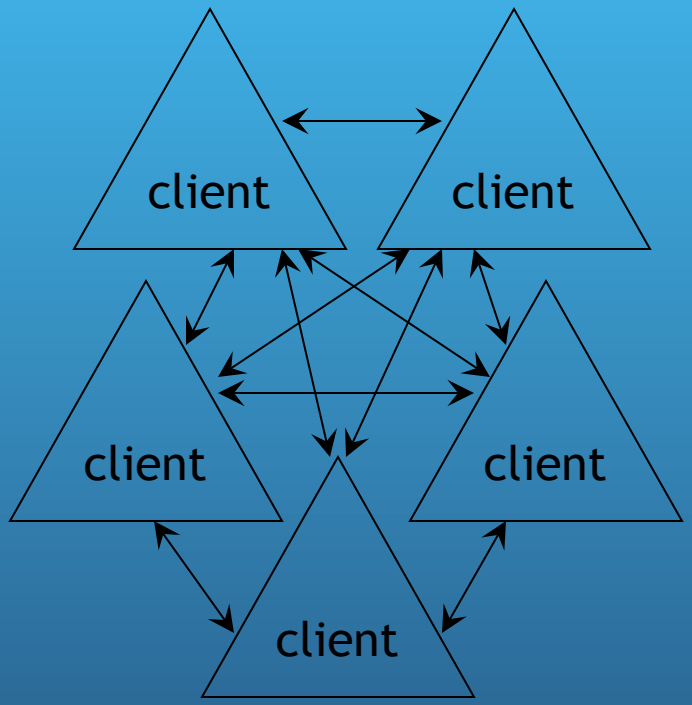
Growth

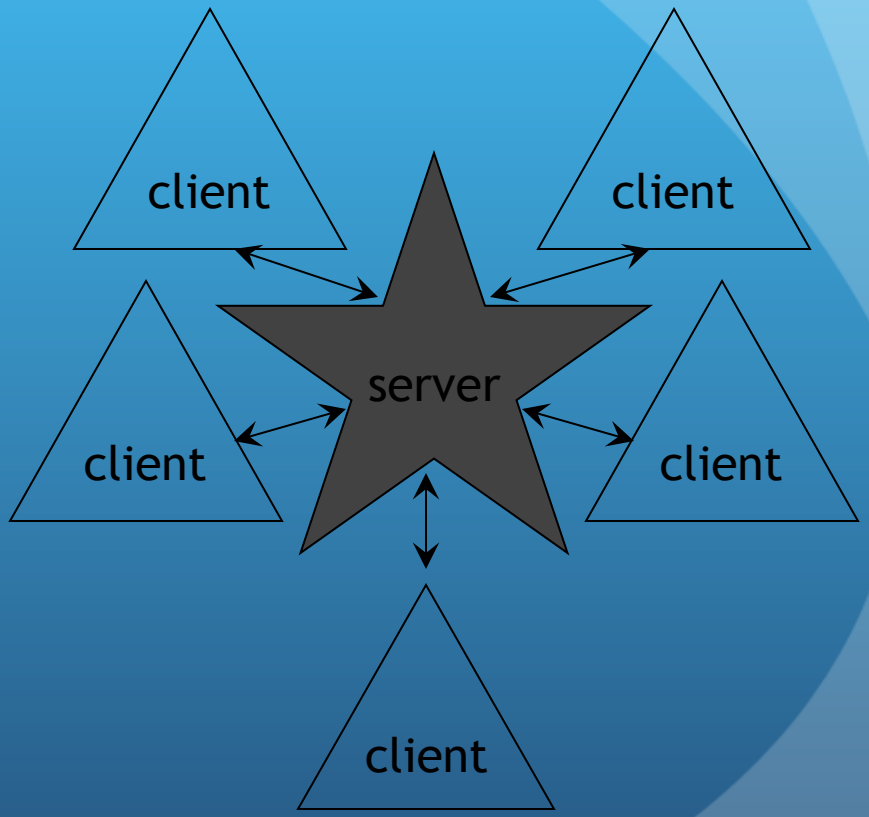

Total MMORPG Subscriptions and Active Accounts listed on this site
Subsciptions in Millions
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
MMOData.net
v3.8

System Architectures

- Client – Server (C-S)
- Peer to Peer (P2P)
- Hybrid – Multiple Servers connected in a meshed P2P configuration and clients connected to one of the servers. This type of configuration may have servers share databases, accounting systems, etc.

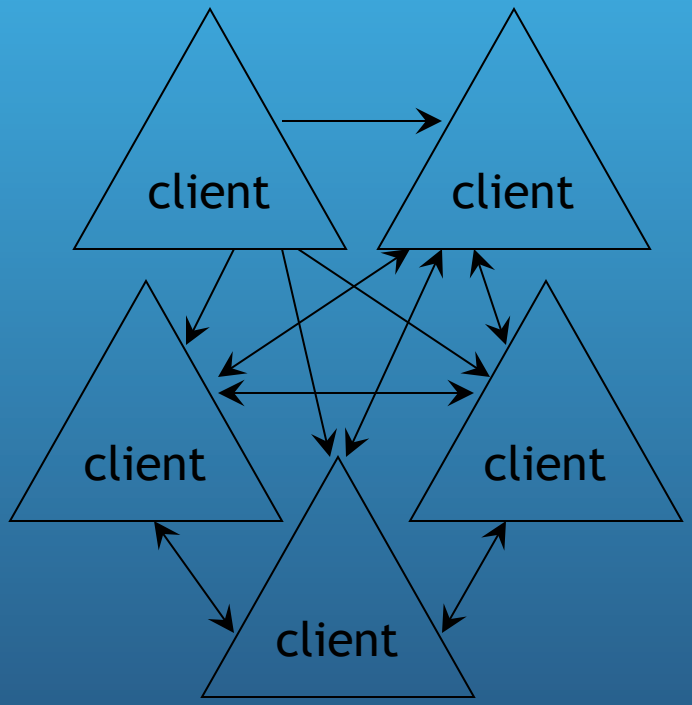
Styles of Networking Architecture



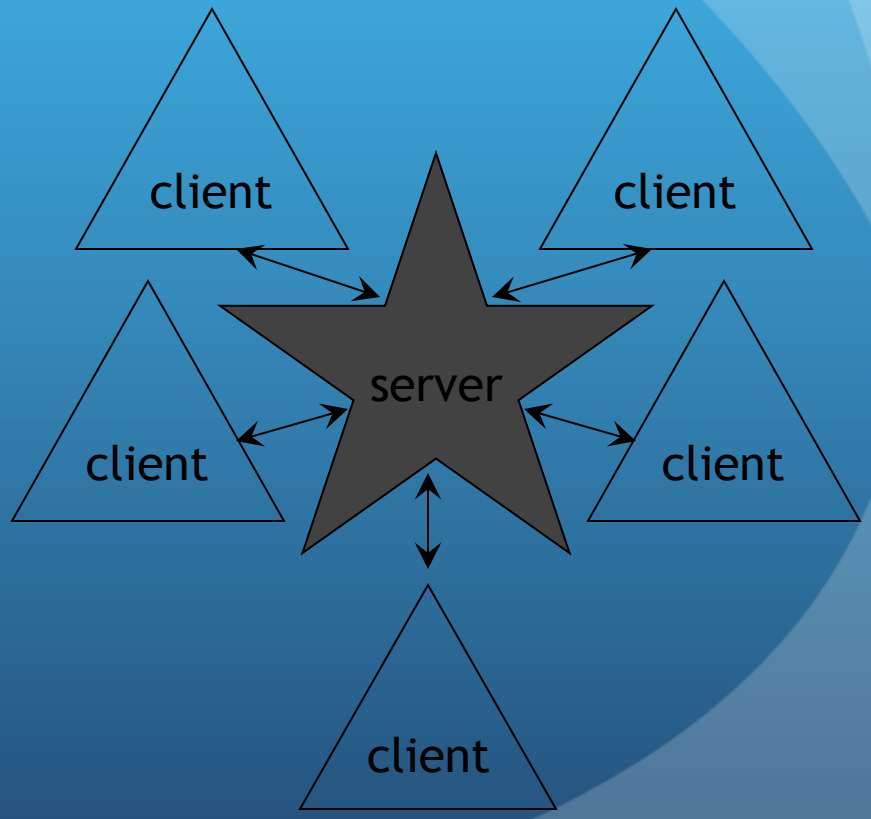


Broadcast

Peer to Peer

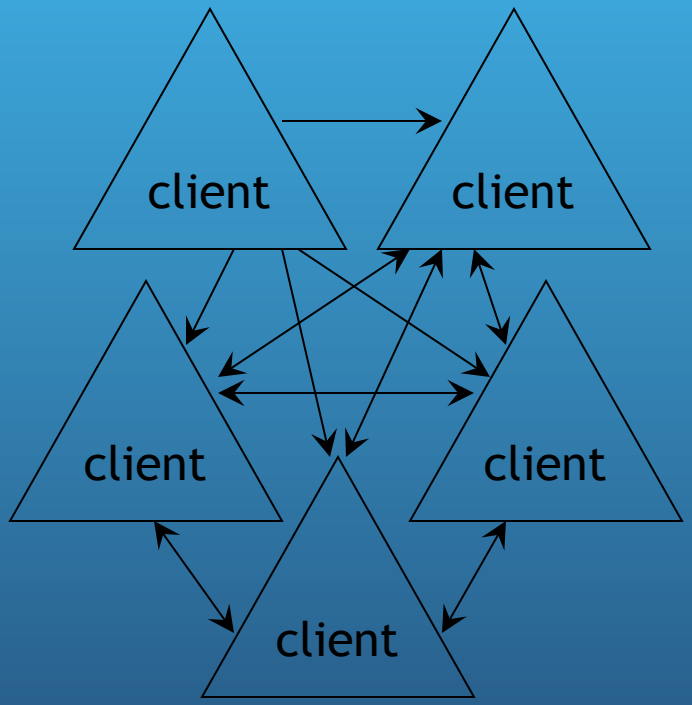


Client - Server

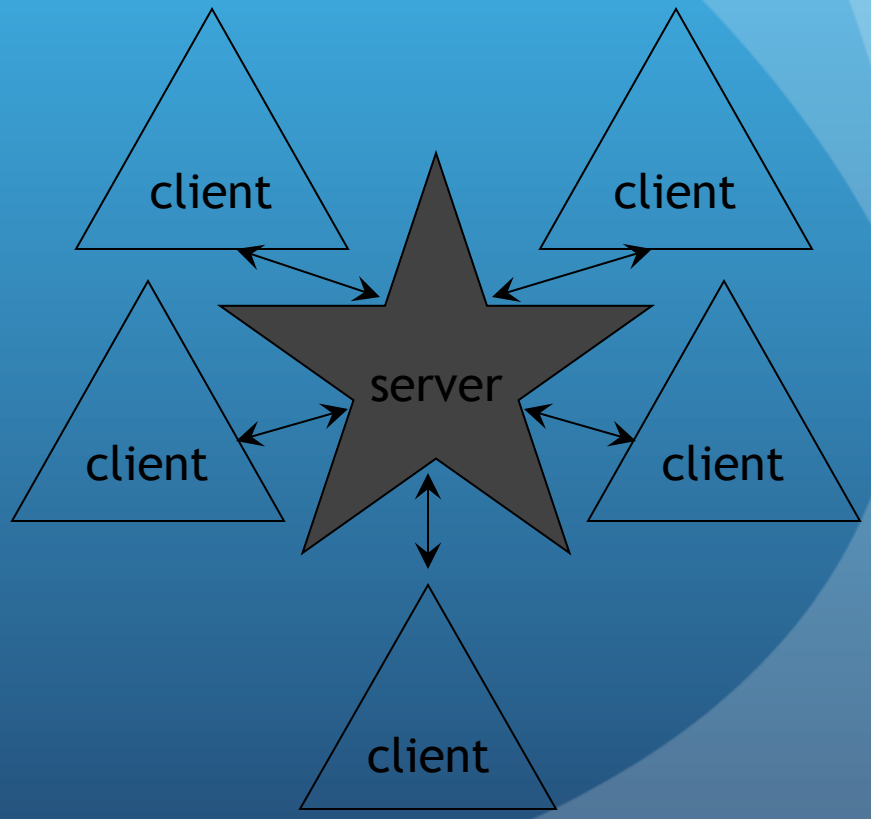


Direct Message

Peer to Peer



Client - Server



Core Domains of MMOGs

- Messaging
- Resource Management
- Persistent Data
- State Management

Messaging

- Transport of messages between nodes (clients/players, servers, databases, etc.)
- Timely arrival of messages is crucial for playability – some games more crucial than for others
- Different system architectures handle messaging differently – in particular the architecture will influence:
  - delivery and
  - ordering
- Protocols have an influence on messaging, but here we only look at the impact of system architecture on messaging mechanism

Resource Management

- Managing resources involves:
  - Resource provisioning
  - Resource allocation
- Provisioning to
  - keep up with demand and
  - to handle spikes/peaks in demand – based on
    - playing time (evenings and weekends),
    - content release
- Allocation to
  - distribute the player load – load balancing
  - clustering based on regions and/or group play

Persistent Data

- Game state is the basis for MMOGs
  - Exists when user is logged on or not
  - Reflects game parameters and player information
  - Constantly updated to reflect user activities
- The system has to store a lot of data to host a game. We classify data as:
  - Static data - not influenced by actual game play once game is activated
  - Dynamic data – changes with time and player activities, interactions, location, etc.

State Management

- Consistent views for all players and the server
  - For large systems it is very hard to maintain due to latencies and losses
  - To manage large state spaces, partitioning is used where only “local interests” are shared with players
- Cheat Prevention – a very serious problem. Players change the state favoring their status.
  - Exploit loop holes in the code – look thru walls, influence latency settings
  - Alter messages – look at content of other messages and change actions based on that
  - Change protocol parameters

Messaging

- Main issue:
  - bandwidth requirements
  - Latency and loss – retransmissions, re-ordering
- C-S vs P2P:
  - C-S more traffic at central server
  - Can be alleviated with hybrid system where regional nodes responsible for distribution of central server updates
  - P2P relies on players sending updates of location, etc., and opens it up to cheating as there is no verification of the state or actions taken and ordering of events

Reducing no. of messages

- The delivery of messages is sped up by disabling ACK messages
- Each message that is sent is delta-encoded and timestamped so that the receiver can look at two messages and determine how much time has elapsed between the sending of the two messages.
- Comparing this to the expected interval between arrivals allows the receiver to detect when messages have not arrived, receiver will then request those messages to be resent.
- The advantage is that delay may be decreased, because the sender does not need to wait for ant AKCs.
- The drawback is that it requires a later message to arrive to actually detect that some messages have been lost. By that time, resending the messages might be a waste, because the data may be no longer relevant.

Area of Interest (AOI)

- AOI is based upon two main ideas.
  - First, the player does not need to know all what is going on in the game, but only that what is relevant to the player.
  - Second the player can only move with a certain velocity.
- Selection of objects in the AOI interest can be done by proximity, aim, and recent interactions.
- Players can abuse this in P2P by not sending relevant AOI data.

Hybrid Architecture Solution

- Central server selects supernodes among the connected users.
- Positional updates: Each supernode is in charge of sending positional updates that occur in an in-game region, to all players inside that region
- Player actions: Central server in charge of any input that can affect the game. The state change is verified and then relayed back to the clients via the supernodes.
- Introducing supernodes, that are in charge of positional updates within a region, does considerably reduce the bandwidth requirements of the central server.
- Letting the central server verify all state changing client input is also a secure way to prevent cheating.

C-S and P2P messaging

- VoroGame is a hybrid architecture in which the world is split into areas based on Voronoi diagrams using the player characters' world location as seeds.
- Each client is in charge of updating objects inside his own cell (region).
- The client estimates for which of his peers the objects in his cell are visible; only those peers will receive client's update messages.
- All player positions are always sent to everyone, so that the partitioning via the Voronoi algorithm can be done locally on each client using the latest positional data.
- Depends on clients deciding who gets data – and what data – open to cheating
- Can be subject to high delays if there is a lot of activity and motion

Ordering

- In C-S systems this is not so difficult to implement. Protocols can be used to ensure in order delivery of all messages.
- In P2P is it more complicated, as clients are responsible for sending updates about their objects and locations to their peers and each player has a different view of the state. Problems arise when there are conflicts – not easy to resolve.
- Hybrid systems have advantage that the central server is in charge of state space. All communication is point to point - only one level up or down for exchange of messages. Protocols can be used to maintain order.

Resource Management - Provisioning

- C-S systems: Provisioning is always a serious problem.
  - Do you increase capacity when user population goes up to meet the demand?
  - But games wane in popularity – demand changes over a period of 6-12months
    - Re-use of resources bought for one particular MMOG for another
    - Hosting of multiple MMOGs on a server, circumvent the “popularity” problem
  - P2P is completely dependent on player resources.

Resource Management - Allocation

- Using server locations to handle allocations of clients – regional distribution is the most common load balancing technique
- Players are also aware of server loads and will pick servers that have open slots and lighter loads
- For P2P, using incentives such as those used in Bit Torrent to encourage users to share their computing power for a group.
- Not all clients are created equal: Implementing algorithms such as a tree of capable client machines that sorts clients capabilities out by situating a client at a particular level in the tree. Pick a client higher up the tree to host.

Persistent Data – Static Data

- Storing game data on user machines has the advantage that it reduces the amount of data which needs to be send in a client-server architecture.
  - In a centralised architecture, it is favourable to send the least amount of information possible to prevent server overload - only data containing actions and movement need to be send.
  - Update distribution a problem for C-S systems – most updates downloaded by 7-80% of players within first 48hrs of release.
  - Using different servers for game content distribution
  - Using P2P mechanisms to distribute new releases and updates.

Persistent Data – Dynamic Data

- Databases – where, how and what to store??
- In C-S systems, to help reduce the load on a database, it is proposed to create a two layer system – upper layer has more static data that does not change as much (like routing tables in a router) and lower layer contains the recent updates (like cache). If a failure occurs, the two are combined to re-create the latest state space.
- In P2P systems, the clients need to maintain a list of "interested" peers and store data related to them. Also only send data to those peers. The list will change depending on actions and proximity, a peer may no longer be an "interested" peer.

State Management - Partitioning

- One way to partitioning an MMOG is by letting several servers run an instance of the game world, e.g., a region, or a map, referred to as a shard.
  - Shards are the result of putting identical copies of an online game onto separate, clustered servers. This enables the game provider to easily and inexpensively ramp up the system to host more players, but it prevents players from interacting with players in other shards.
  - Shards are not connected to each other; a player logs into one specific shard. It's player data is stored only in that shard's database and the player only interacts with other players in that specific shard. A shard has a limit in the number of players it can support, so when the number of players grows, they have to queue.
- Another partitioning method is to have servers only run regions of the game (different from geographic regions). Then users can easily update their state with other players in the region only.

State Management - Cheating

- A huge topic – beyond today's discussion!

Reference

- A Survey on MMOG System Architectures, Mattijs Driel, Jos Kraaijeveld, Zhi Kang Shao, Remco van der Zon {M.G.Driel, J.M.Kraaijeveld, Z.K.Shao, R.W.L.vanderZon}@student.tudelft.nl, March 30, 2011