Seminar aus Informatik
Universität Salzburg
SS 2009
LV-Leiter: Wolfgang Pree

Milena Trujic,
Schönegger Andreas

14. Juli 2009

Zusammenfassung

In diesem Paper behandelt unser Team das Thema Massively Multiplayer Games. Da solche Arten von Spielen sich einer konstanten Steigerung an Beliebtheit erfreuen ist es logisch Forschung in diesem Bereich zu betreiben. Massively Multiplayer Games haben enorme Anforderungen an Hardware und Software und die Menge an Spielern handhabbar zu machen. Unser Paper beginnt mit einer Definition und den technischen Anforderungen an die Spiele. Im zweiten Teil beschäftigen wir uns dann mit den Architekturen. Wir stellen gängige Netzwerkarchitekturen vor und kommen dann zu speziell für MMGs ausgelegten Architekturen. Zum Schluss kommen wir dann noch zu einem Beispiel für MMGs.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 DEFINITION MASSIVELY MULTIPLAYER GAMES (MMGS)

Bei Massively Multiplayer Games handelt es sich um Computerspiele, wo eine Vielzahl an Teilnehmern simultan, über ein Netzwerk, interagiert. In Abhängigkeit von den Spielcharakteristiken, werden unterschiedliche Netzwerkarchitekturen für die Spiele verwendet.

2 TECHNISCHE ANFORDERUNGEN FÜR MMGS

Die Erwartungen kann man in 5 grobe Anforderungen aufteilen:

- Sicherheit
- Konsistenz
- Skalierbarkeit
- Netzwerkbandbreite

Die Anforderungen werden im Folgenden etwas genauer erläutert.

2.1 Sicherheit

Bei netzwerkbasierten Computerspielen muss gewährleistet sein, dass man den Berechnungen der Spielzüge trauen kann. Dem Spieler darf keine Möglichkeit gegeben werden, das Spiel zu manipulieren und an verborgene Informationen zu gelangen (Cheating).

2.2 Konsistente Spielwelt

Da sich alle Spieler in einer gemeinsamen Spielwelt befinden, muss jeder Spieler eine exakte Kopie der Spielwelt besitzen. Die Verteilung der Informationen an die Spieler muss so ablaufen, dass bei Ausfall einiger Rechner die Daten dennoch nicht verloren gehen. Sollte die konsistente Welt nicht mehr aufrecht erhalten werden können, muss das System möglichst schnell auf diese Veränderung reagieren.

2.3 Skalierbarkeit

Hier geht es um die Performanz des Spieles, bei wachsenden Eingabemengen bzw. steigender Spielerzahl. Ein Spiel ist gut skaliert wenn der Ressourcenverbrauch möglichst linear zur Anzahl der Teilnehmer und Ereignisse in der Spielwelt steigt.

2.4 Netzwerkbandbreite

Die Netzwerkbandbreite ist das Datenvolumen, welches in einer bestimmten Zeit über ein Netzwerk übertragen werden kann. Neue Spieler sollten die Spielwelt betreten, ohne dass es bei anderen Spielern zu Verzögerungen kommt. Die Verzögerungszeit (Latenz) ist sehr wichtig, da der Spieler möglichst schnell auf Veränderungen reagieren muss und somit die Information so rasch wie möglich benötigt.

3 TYPISCHE ARCHITEKTUREN

3.1 Client Server Modell

Die meisten MMGs benutzen dieses Modell auf Grund der Einfachheit. In der klassischen Client-Server Netzwerkarchitektur gibt es einen Server, der einen Dienst zur Verfügung stellt. Dieser Dienst kann dann durch mehrere Clients in Anspruch genommen werden.

Jeder Client verbindet sich direkt zum Server, welcher Informationen, individuell, zu jedem Client, auf Anfrage, übermittelt.

Ein Spieler beschließt als Server zu fungieren und die restlichen Spieler verbinden sich direkt zu seinem Rechner. Bei jeder Bewegung im Spiel, z.B. schießen, laufen oder einer anderen Aktion wird diese Information zum Server gesendet, welcher das Resultat berechnet und an die übrigen Teilnehmer weiterleitet.

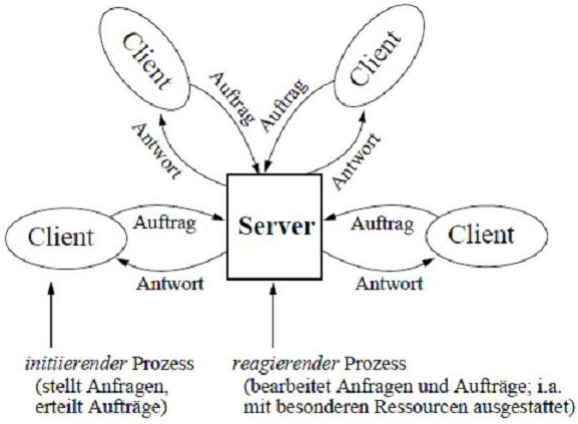

Figure 1: Client Server Architektur

3.1.1 Vor- und Nachteile der Client Server Architektur

Die Vorteile sind, dass es wenige Konsistenzprobleme gibt, da alles vom Server berechnet wird. Zudem ist diese Architektur einfach zu administrieren, da sich

alle Teilnehmer auf einem Server aufhalten.

Der Server verhindert unberechtigten Zugriff auf Informationen und reduziert somit die Gefahr des Cheating.

Die gesamte, für das Spiel notwendige, Kommunikation läuft über den Server ebenso wie die benötigten Dienste. Fällt der Server aus, kann auf das ganze System nicht mehr zugegriffen werden. Ein weiterer Nachteil dieser Architektur ist die Bandbreite, da es rasch zu einem Bottleneck kommen kann.

3.2 Client-Multi-Server Architektur

Da ein einziger Rechner aber nicht in der Lage ist mehr als 1000 Spieler aufzunehmen, wird typischerweise für MMGs eine Client-Multi-Server Architektur verwendet.

Der „Server“ ist eine Gruppe von Rechnern denen unterschiedliche Aufgaben zugeteilt wurden. Jeder Rechner hat eine unterschiedliche Verantwortung im Spiel. Die Gruppen können entweder als Cluster oder Grids aufgebaut sein.

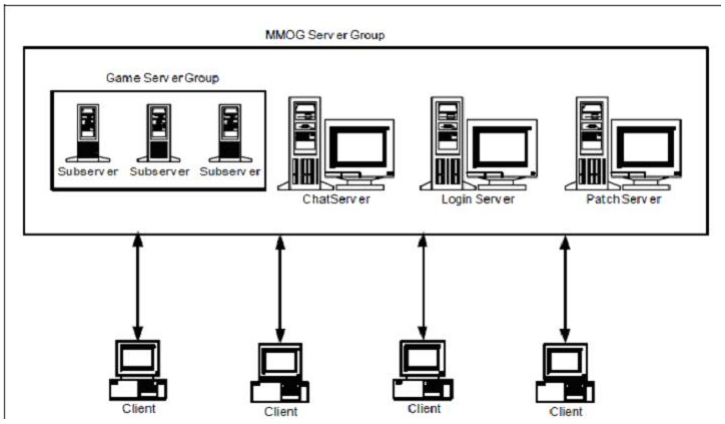

Figure 2: Client Multi Server Architektur

Wenn ein Client versucht sich zum Spiel anzumelden, verbindet er sich mit dem Login Server, dieser überprüft die Daten des Clients (existiert der Benutzer?, bezahlt?; ...). Später wird der Client zum Patch Server weiterverbunden um zu überprüfen welchen Spielstatus der Client besitzt und eventuell ein Update durchzuführen. Erst danach wird der Spieler zum Game Server verbunden und eventuell zu einem zusätzlichen Chat Server.

In der oberen Grafik ist der Game Server ebenfalls auf mehrere Unterserver aufgeteilt, z.B. betreut jeder einen bestimmten geografischen Bereich der Spiel-Welt.

3.2.1 Vor- und Nachteile der CMS Architektur

Fällt ein Server aus, so kann trotzdem auf das System zugegriffen werden. Leider entstehen auch Nachteile. Einer davon ist die mögliche Inkonsistenz, da die Ar-

chitektur mit mehreren parallelen Simulationszuständen umgehen muss. Ebenso werden mehrere Server und ein Netzwerk, welches diese verbindet, benötigt und das macht die Architektur Kostenintensiv.

3.3 Peer-to-Peer Architektur

In einem Peer-to-Peer Netzwerk sind alle Teilnehmer gleichberechtigt. Das heißt, dass ein Teilnehmer Dienste zur Verfügung stellt und auch von anderen in Anspruch nimmt. Eine klare Rollenverteilung, wie in der Client-Server Netzwerkarchitektur existiert somit nicht.

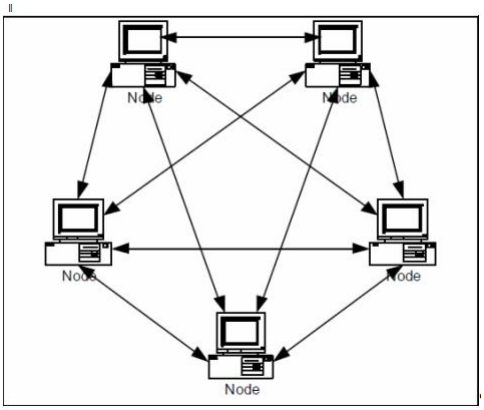

Figure 3: Peer to Peer Architektur

3.3.1 Vor- und Nachteile der Peer-to-Peer Architektur

Der offensichtlichste Vorteil dieses Modells ist, dass kein zusätzlicher Server benötigt wird. Dies reduziert den Hardwareaufwand. Weiters wirkt sich der Ausfall eines nicht gravierend aus, die nicht betroffenen Spieler können weiter spielen.

Mit steigender Peer-Anzahl wächst der Verbindungsaufwand. Dies bewirkt, dass Peer-to-Peer Modelle mit Unicast-Verbindungen nur begrenzt skalierbar sind. Bei Verwendung von Broadcast/Multicast-Verbindungen ist die Anzahl der Verbindungen gleich der Anzahl der Peers, jedoch muss dann jeder Peer alle Nachrichten verarbeiten, auch wenn sie für ihn nicht relevant sind. Da es keinen zentralen Server gibt, müssen die Teilnehmer in diesem Modell sich selbst um das Finden im Netzwerk kümmern. Ein Schutz vor Cheatern oder Hackern ist somit nicht bzw. nur sehr schwer möglich.

3.4 Peer-to-Peer mit Central Arbiter Architektur

Bei dieser Architektur tauschen die Spieler Updates nicht nur untereinander aus, sondern senden diese auch zum Central Arbiter. Die Aufgabe eines Central

Arbiters ist es, auf Updates der Spieler zu lauschen, den globalen Status des Spieles zu simulieren und auf Inkonsistenz zu achten. Treten keine Inkonsistenzen auf, verhält sich der Arbiter ruhig und läuft im Hintergrund, er sendet keine Nachtichten an die Spieler. Erkennt der Arbiter aber eine Inkonsistenz, so wird er diese beheben, ein neues Update erzeugen und dies an alle Spieler übermitteln. Die Spieler bei denen Korrekturen nötig waren, sollten zu dem vorher akzeptierten Zustand zurückkehren.

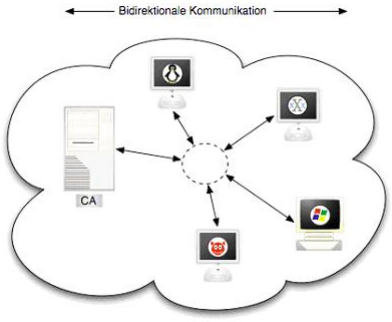

Figure 4: PP-CA Architektur

3.4.1 Vor- und Nachteile der PP-CA Architektur

Diese Architektur vereinigt alle Vorteilte der Peer-to-Peer Architektur und der Client-Server Architektur. Der Unterschied zwischen der PP-CA und der CS Architektur ist aber, dass der Arbiter nur korrigierte Updates an die Spieler sendet und sonst im Hintergrund läuft, wo hingegen der Server zu jeder Update Periode zu allen Spielern Nachrichten übermittelt.

Somit sind bei der PP-CA Architektur die Bandbreiteanforderungen um einiges geringer. Der Nachteil der klassischen P2P-Architektur, nämlich das Sicherheitsproblem bezüglich Cheating, wird ebenfalls durch den Arbiter beseitigt.

Der Nachteil bei dieser Architektur könnte das Fehlen von Rechnerleistung sein, wenn zu viele Spieler gleichzeitig am Spiel teilnehmen. Falls Inkonsistenz auftritt, wird natürlich auch etwas mehr Bandbreite benötigt.

4 Spezielle Architekturen:

Im folgenden Kapitel werden wir jetzt eine spezielle Architektur und einen Algorithmus zum Beherrschen solcher Architekturen vorstellen. Die Hauptidee bei den meisten Architekturen oder Algorithmen ist es, das Spiel in Bereiche zu unterteilen. Dieser Trick wird durch drei Eigenschaften ermöglicht, die in den meisten MMOPRGs vorhanden sind. Die drei Regeln auf die wir aufbauen sind: dass ein Spieler sich nur mit begrenzter Geschwindigkeit fortbewegen kann die

Sicht des Spielers auf einen Teil des Spielfeldes begrenzt ist nur Daten aus der näheren Umgebung wichtig sind für das aktuelle Spielgeschen. Aus diesen Eigenschaften kann man nun ableiten, dass es möglich ist das Spiel aufzuteilen und somit das Übertragungsvolumen und die Rechenleistung der einzelnen Knoten zu verkleinern. Das Aufteilen des Spielfeldes ist laut unseren Nachforschungen das gängigste und zurzeit am meisten eingesetzte Verfahren, um den enormen Anforderungen an die Architektur Herr zu werden.

4.1 Mirrored Arbiter Architektur

Als erstes werden wir eine mögliche Architektur für diese Regionsunterteilung untersuchen. Diese Architektur wurde vorgestellt in dem Paper „Mirrored Arbiter Architecture – A Network Architecture for Large Scale Multiplayer Games". Die Autoren dieses Papers sind Lan Yang and Peerapong Sutinrerk.

Die Idee hinter den Mirrored-Arbiter (MA) ist zwei gängige Architekturen zu kombinieren um Synergien der Beiden zu nutzen. Im Prinzip ist es eine Kombination der bereits erklärten Architekturen PP-CA und CMS. Für jede Region im Spiel die wir unterteilen verwendet man die PP-CA Architektur. Um jetzt die einzelnen Regionen zu verbinden verwendet man die CMS Architektur.

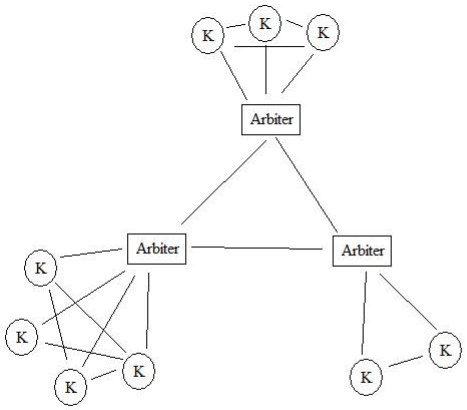

Figure 5: Mirrored Arbiter Architektur

Die Kreise stellen die Klienten dar. Alle Klienten sind mit allen anderen Klienten, die in derselben Region sind, verbunden. Außerdem hat jeder Klient eine Verbindung zu dem Arbiter seiner Region. Die Kommunikation in einer Region läuft über Multicasts. Durch die Verwendung von Multicasts kann die benötigte Bandbreite sowohl auf Server als auch auf Klientseite reduziert werden. Bei der Kommunikation zwischen den Arbitern benutzt man ein P2P network dass alle Arbiter miteinander verbindet. Die Arbiter benutzen Interest Management um zu entscheiden wann ein Knoten seine Region betritt oder

verlässt. Verlässt also ein Knoten den Bereich eines Arbiters wird dieser zum nächsten Arbiter weitergereicht und in das dortige Netzwerk integriert.

4.1.1 *Algorithmus des MIT*

Die zwei Hauptprobleme die durch das Aufteilen der Welt entstehen sind, Korrektheit und serverübergreifende Aktionen. Im Folgenden werden wir uns mit dem am Massachusetts Institute of Technology entwickelten Algorithmus beschäftigen der sich mit dieser Problematik auseinandersetzt. Die Daten und Tests haben wir aus dem Paper „A Distributed Architecture for MMORPG“ entnommen. Das Paper wurde von Marios Assiotis und Velin Tzanov verfasst. Wir werden nicht den ganzen Algorithmus behandeln sondern nur die Teile die sich konkret mit den oben genannten Problemen beschäftigen um eine mögliche Lösung zu demonstrieren. Um den Algorithmus verstehen zu können benötigt man zwei im Paper erklärte Konzepte. Den sogenannten „Locking Mechanism“ und den „Event Announcing Mechanism“. Beim Locking Mechanism unterscheidet man zwei Arten von Locks. Den Region Lock und den Object Lock. Beim Region Lock werden geographische Teile der virtuellen Welt gesperrt. Die Verantwortung einer solchen Sperre liegt immer beim Server der die Region verwaltet. Der Objekt Lock dagegen sperrt nicht ein Gebiet sondern ein Objekt. Als nächstes behandeln wir den Eventauslöse Mechanismus. Im Normalfall wird ein Event von dem Server behandelt in dessen Region es sich befindet. Ist ein Event jedoch an der Grenze eines Servers und überschreitet diese, wird die Eventbehandlung dennoch vom dem Server der es ausgelöst hat durchgeführt und nicht direkt auf dem Server auf dem es sich jetzt befindet abgehandelt. In der folgenden Grafik haben wir eine Illustration dieser Situation. Sobald sich P1 in Reichweite von P2 befindet würde auf Server 2 ein Event ausgelöst. Dieses Event müsste eigentlich von Server 1 behandelt werden, da es ja in seinem Zuständigkeitsbereich liegt. Tatsächlich wird jedoch das Event von Server 2 ausgelöst und behandelt.

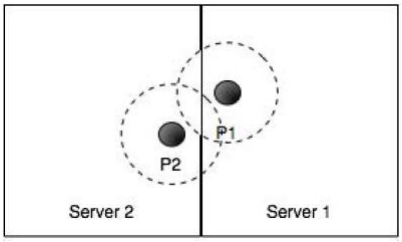

Figure 6: Grenznahe Events

Nun widmen wir uns der Lösung unserer vorher genannten Probleme. Das erste Problem ist wenn ein Event mehrere Server betrifft. Nehmen wir an, dass Server 2 (S2) der auslösende Server ist. Das Event betrifft jetzt jedoch nicht nur diesen Server sondern auch S1 und S3.

- S2 fragt an um einen Region Lock auf das betreffende Gebiet und ein Object Lock auf alle betreffenden Objekte.
- Server S2 führt den Event aus und benachrichtigt S1 und S3 wie sie ihren Status anpassen sollen.
- S2 gibt alle Locks Frei
- Nun erhalten alle Klienten das für sie benötigte Update

Als nächstes kommen wir zu dem transportieren von Objekten.

- Schritte von Server 1:
  - Anfragen um Regionslocks bei S2 und bei sich selbst
  - Objektlock auf das entsprechende Objekt anfragen
  - Die ID der letzten Aktion wird an Server 2 gesendet
  - S1 startet den Transfer der Statusdaten an S2
  - S1 gibt alle Locks Frei
- Schritte von Server 2:
  - Ausführen der angefragten Regionslocks von S1
  - Speichern der von S2 gesendeten ID
  - Erhalten der Statusdaten von S1

Zum Abschluss wollen wir noch ein paar Performance Tests darstellen um die Scalability des Algorithmus zu zeigen.

Als erstes wir die Performance eines Einzelservers mit 100, 150 und 200 Klienten im Vergleich zu einer Architektur mit 3 Servern die den MIT Algorithmus verwenden mit 200, 300 und 400 Klienten dargestellt. Man kann sehen, dass die Leistung des Verteilten Systems besser ist als die des Einzelsystems.

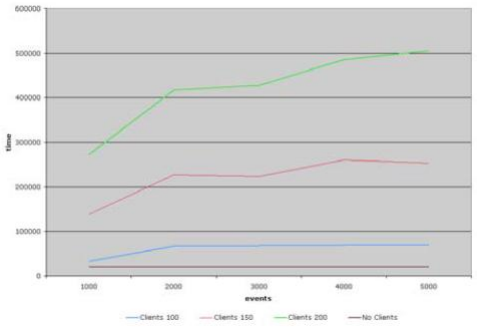

Figure 7: Einzelserver mit 100, 150 und 200 Kienten

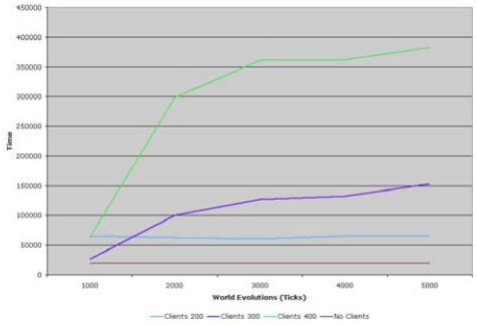

Figure 8: 3 Server mit 200, 300 und 400 Klienten

Als zweiten Test möchten wir noch zeigen dass auch bei viel grenznahen Aktivitäten die Performance noch in einem guten Bereich abläuft. Hierfür vergleichen wir die Leistung von zufällig auftauchenden Klienten zu Klienten die in Grenzbereichen auftauchen.

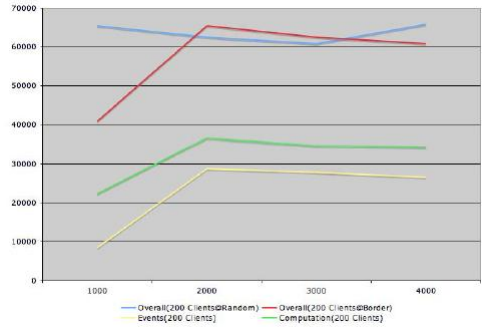

Figure 9: Grenzaktivitäten

5 Beispiel eines MMGs: World of Warcraft

World of Warcraft ist ein Fantasy–MMORPG. Es wird auf CD-s ausgeliefert und muss auf dem Rechner installiert werden, diese Rechner fungieren dann als Clients.

5.1 Server

Die allgemeine Serverstruktur ist auf die ganze Welt verteilt, z.B. USA, Europa und Asien. In jeder dieser Lokalitäten gibt es jeweils eine Serverfarm. Innerhalb dieser Farmen wird pro Server eine Parallelwelt gefahren, die hier auch Realm genannt werden.

5.2 Sicherheit

Es gibt ein Team von ca. 100 Leuten, das sich aktiv und rund um die Uhr darum kümmert, dass das Spiel am Laufen bleibt. Es leistet technischen Support und Game–Balancing. Dieses Team führt dann auch die regelmäßigen Wartungsarbeiten aus.

5.3 Die Welt von World of Warcraft:

Alle Spieler spielen auf einer einzigen riesigen Weltkarte, mit verschiedenen Städten, Dörfern und sogar Wüsten und Wäldern usw.

5.4 Der Charakter:

Am Anfang der Erschaffung des Avatars muss der Spieler eine Entscheidung treffen, er muss sich für eine von zwei Seiten entscheiden, entweder Horde oder Allianz. Danach muss er nur noch auswählen zu welchem von den acht Völkern und neun Klassen der Avatar gehören soll. Diese Entscheidungen bestimmen, welche Fähigkeiten der Charakter später lernen kann und wer seine Freunde und seine Feinde sind.

5.5 Spielprinzip:

Der Spieler sammelt Erfahrungspunkte in dem er sogenannte Quests annimmt bzw. einen Kampf gegen computergesteuerte Monster gewinnt. Weiters gibt es Erfahrungspunkte für das Erkunden neuer Gegenden bzw. Städte. Nach einer bestimmten Anzahl von Erfahrungspunkten kommt der Spieler bzw. der definierte Charakter ein Level weiter.

5.6 Kommunikation

Die spielinterne Kommunikation wird durch einen internen Chat–Client geregelt. Es können auch externe Chatprogramme wie ICQ oder Skype verwendet werden, diese sind jedoch nicht im Spiel integriert. Die Kommunikation ist durch das komplette Spiel möglich.

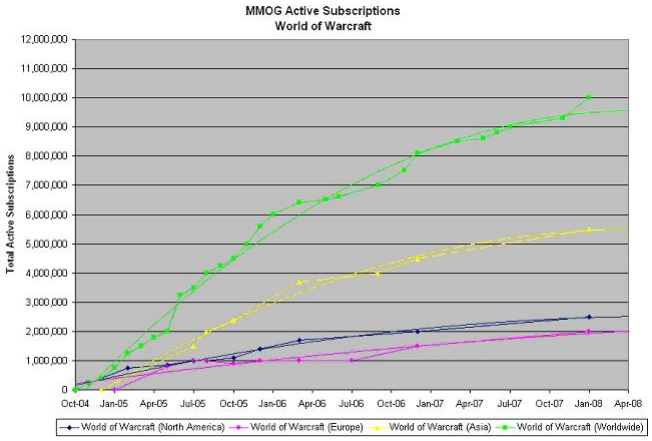

Figure 10: Statistik über die Spieleranzahl beim World of Warcraft

References

[John Moores] Agents-Based Modeling for a Peer-to-Peer MMOG Architecutre, Abdennour el Rahalibi, Madjid Merabti, Liverpool John Moores University (2005)

[MA] Mirrored Arbiter Architecutre – A Network Architecure for Lage Scale Multiplayer Games, Lan Yand, Peerapong Sutinrerk, California State Polytechnic University

[MIT] A Distributed Architecture for MMORPG, Marios Assiotis, Velin Tzanov (2006)

[1] http://www.wow-europe.com/de/index.xml

[2] Abbildung 1: http://www.vs.inf.ethz.ch/edu/WS0304/VS/slides/Vorl.VertSys03_04_5c.pdf

[3] Abbildung 3: http://www.semanticmedia-showcase.de/MuseumSM/MMORPGS/Peer2Peer.htm

[10] Abbildung 10: www.mmogchart.com/charts