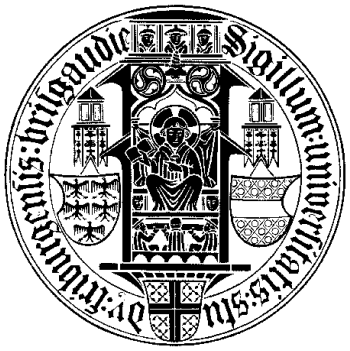
Ausarbeitung im Rahmen des Seminars Peer-to-Peer Netzwerke

Personal Power Plant

Amir Alsbih



2007

Betreuer: Arne Vater

Inhaltsverzeichnis

1 Einleitung

Kann man Peer-to-Peer-Netzwerke wirklich nur zum Download von Copyright geschützten Materialien verwenden? In meiner Ausarbeitung verfolge ich das Ziel das Potential von Peer-to-Peer Netzwerken in der Verwendung von Verteiltem Rechnen aufzuzeigen. Dabei wird zunächst die Theorie von Peer-to-Peer-Netzwerken erklärt um zur Verwendung von Peer-to-Peer Netzwerken, für das Verteilte Rechnen an dem konkreten Beispiel von P3 - Personal Power Plant aufzuzeigen. Das Projekt Personal Power Plant eignet sich hervorragend um die Möglichkeiten von Peer-to-Peer Netzwerken aufzuzeigen, als auch um die Problematiken, die sich durch die Verwendung von Verteiltem Rechnen im Internet ergeben aufzuzeigen.

1.1 Peer-to-Peer Netzwerke

Ein Peer-to-Peer Netzwerk bezeichnet ein Netzwerk aus gleichberechtigten Partnern (im folgenden Knoten genannt), in dem jeder Knoten Dienste zur Verfügung stellen oder nutzen kann. Das Netzwerk wird dabei ohne eine zentrale Steuereinheit aufgebaut. Ein großes Problem dass durch diese Architektur resultiert, ist dass fehlen von vertrauensvollen Knoten. Jeder beliebige Knoten, kann das Netzwerk zu jedem beliebigen Zeitpunkt betreten oder auch verlassen ohne das die Hintergründe oder Informationen über die Absichten des Knoten bekannt sind.
Der hohe Stellenwert von Peer-to-Peer Netzwerken wird an der Tatsache deutlich, dass die Peer-to-Peer Kommunikation derzeit mehr als 50% der Internetkommunikation ausmacht.[Schi06]

1.2 Was ist Verteiltes Rechnen

Verteiltes Rechnen, bezeichnet die Aufteilung rechenintensiver Anwendung auf mehrere Computer. Der Gedanke ist dabei, auf Großrechner zu verzichten, die imense Kosten verursachen und an deren Stelle, die ungenutzten Kapazitäten vieler Heim-PCs zu nutzen.
Nimmt man z.B. die kryptographische Analyse eines Textes, so würde man Rechner Eins, das Verfahren A auf den Text anwenden lassen, Rechner Zwei, das Verfahren B, ... die Ergebnisse dieser Analysen werden anschließend beim Projekt-Leiter-PC zusammengeführt und verwertet.
Dieser Ansatz wird in vielen Bereichen der Wissenschaft genutzt, wie z.B. in der:

1. Astronomie: SETI@home - Sucht nach Signalen außerirdischer Intelligenzen.

2. Klimatologie: ClimatePrediction.net - Erprobt Klima vorhersagen für Zeiträume von 50 bis 100 Jahren.

3. Biologie, Chemie, Medizin: FightAIDS@Home - Sucht nach neuen Medikamenten zur Behandlung HIV-Infizierter.

4. Physik, Technik: CuboidSimulation - Projekt zur Entschlüsselung der Statistik asymmetrischer Würfel.

5. Mathematik: PrimeGrid - Faktorisiert RSA-640.

6. Nanotechnologie: Spinhenge@home - Ist ein Projekt der FH Bielefeld, das die Spindynamik in magnetischen Molekülen untersucht.

[Proj]

2 Personal Power Plant

2.1 Entstehungsgrund für Personal Power Plant

Die bisherigen Projekte für Verteiltes Rechnen, welche die Rechenkapazitäten von Internetanwendern nutzen, verletzen jedoch den eigentlichen Grundgedanken von Peer-to-Peer Netzen[Schi06], denn die Ressourcen werden hierbei nur vom Projekt-Eigentümer genutzt, die Ressourcenprovider ziehen keinen Nutzen daraus.
Personal Power Plant realisiert Verteiltes Rechnen unter dem Ansatz von Peer-to-Peer, durch die Eigenschaft dass jeder Knoten des Netzwerks die Ressourcen der anderen Knoten nutzen kann, indem er einen Job einträgt.[ShTS05]

2.2 Personal Power Plant als Erweiterung von JXTA

Personal Power Plant greift auf die JXTA Bibliothek von SUN zurück, um die Peer 2 Peer Funktionalitäten, wie die Suche nach Peers,Projekten, . . . zu realisieren, als auch um das Netzwerk-Overlay zur erstellen.[ShTS05]
Die einzelnen Rechner im JXTA-Overlay nennen sich Peers und zeichnen sich durch eine eindeutige PeerID aus. Über diese PeerID erfolgt die Kommunikation im JXTA-Netzwerk-Overlay.
Da die JXTA Bibliothek eine allgemeine Bibliothek für Peer-to-Peer Anwendungen ist [JXTA], bietet die API komplexe Aufrufe und Optionen, welche für Anwendungen, die sich mit Verteiltem Rechnen beschäftigen irrelevant sind. Aus diesem Grund und von dem Gedanken ausgehend, die die Programmierung von

parallelen Algorithmen und verteiltem Rechnen zu vereinfachen, in dem der Programmierer sich auf den Algorithmus und das Problem konzentrieren kann, wurde Personal Power Plant geschaffen.

2.3 Funktionsweise von Personal Power Plant

Ein Projekt für verteiltes Rechnen entspricht einer Job-Gruppe in Personal Power Plant, der Anwender trägt also einen Job mit einer Beschreibung ein, und erstellt damit eine Job-Gruppe mit einer eindeutigen JobID innerhalb der JXTA Basis Gruppe, wie man in der Abbildung 1 sieht. Die Peers in der Basis Gruppe können

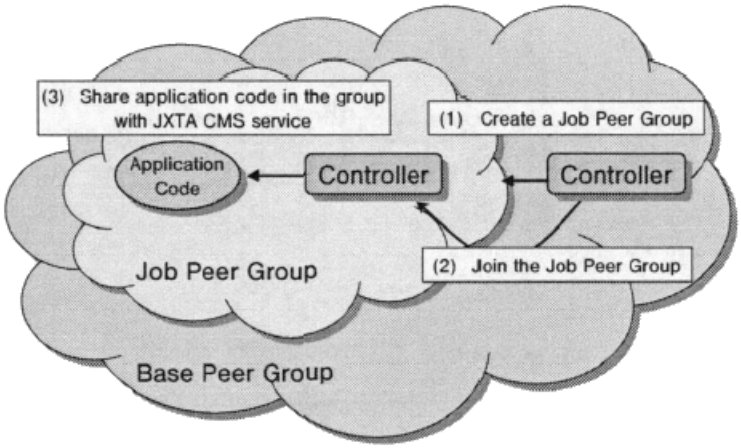


Abbildung 1. Joberstellung in P3

un einen Beitrag zu diesem Projekt in Form von Rechenleistung leisten, in dem sie der Job-Gruppe beitreten.
Das Eintreten in einen Job erfolgt dabei automatisch anhand eingestellter Kriterien wie dem Jobersteller, dem Projektnamen und Schlüsselwörtern, sofern der Personal Power Plant User über keine Graphische Oberfläche verfügt, hat der User hingegen eine Graphische Oberfläche, so wählt er die Job-Gruppen, in die er eintreten möchte manuell aus, eine Automatik, ist für diesem Fall nicht vorgesehen. Ein Peer kann dabei in beliebig vielen Gruppen gleichzeitig sein und sowohl Ressourcen anbieten, als auch Projekte starten.
Wie aus der Abbildung 2 hervorgeht, gibt es Knoten die nur ein Projekt unterstüzen, oder Knoten - wie den mittleren Knoten - die in zwei Projekten gleichzeitig sind, oder auch Knoten, die noch kein Projekt unterstützen und deshalb nur in der Base-Gruppe sind.

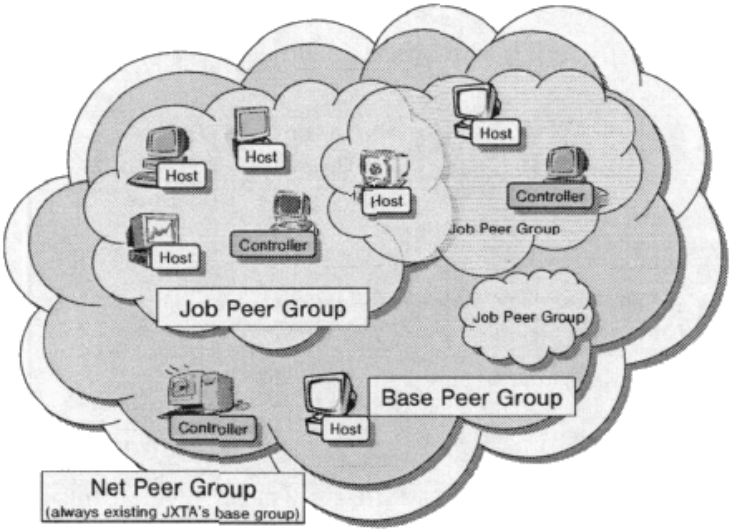


Abbildung 2. Gruppen / Projektübersicht in P3

2.3.1 Personal Power Plant Projekt im Detail

Personal Power Plant Projekte bestehen aus dem Java-Quellcode welcher als JAR Datei compiliert wird und den Daten, welche bearbeitet werden sollen.
Ein Host, der Ressourcen für das Projekt bereitstellen möchte, tritt als erstes der Gruppe bei, lädt den Job herunter und überprüft dann die digitale Signatur, ist diese korrekt beginnt er mit der Arbeit. Der Code kann dabei in einer abgeschirmten Umgebung ausgeführt werden (Sanbox), in der kein Schaden auf dem Host-System verursacht werden kann oder aber auch als normale Implementation bereitgestellt werden. Ein Peer führt diese Schritte, die in Abbildung 3 graphisch dargestellt sind, für jedes Projekt durch, das er unterstüzen möchte. Es gibt dabei keine Beschränkung auf eine bestimmte Anzahl von Projekten, welche ein Peer gleichzeitig unterstützen kann. [ShTS05]

Bedingt durch die immer häufiger anzutreffenden Schadensroutinen also Programme deren eigentliches Ziel es ist die Clienten mit Würmern, Trojanern oder Rootkits zu infizieren und damit eine Kontrolle über die Rechner zu bekommen, werden Projekte, welche in einer abgeschirmten Umgebung (Sandbox) ausgeführt werden können, häufiger Unterstützt. Durch die Verwendung einer Sandbox muss sich ein Peer, welcher Kapazitäten bereitstellt keine Gedanken um seine Systemsicherheit machen muss.
Damit die Entwickler sich ganz auf die Problemstellung konzentrieren können und sich keine Gedanken um die Verwaltung und die Verteilung von Workunits machen müssen, stellt Personal Power Plant drei Bibliotheken für die Entwicklung von parallelen Applikationen bereit:

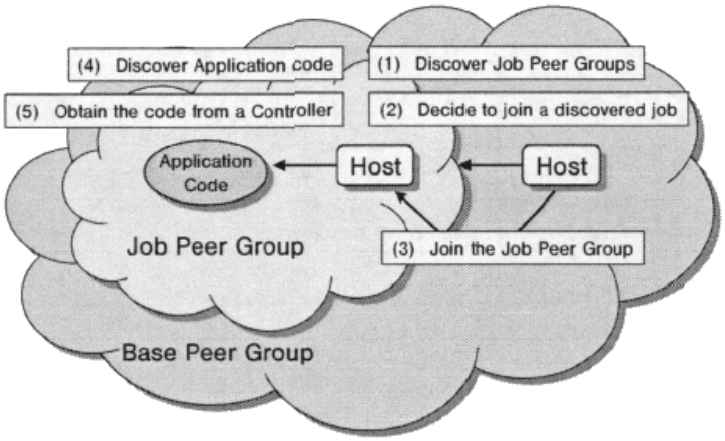


Abbildung 3. Projektbeitritt in P3

1. Object Passing Library: Diese Bibliothek setzt direkt auf der komplexen JXTA-API auf, vereinfacht ihre Methoden und bietet damit letztenendes eine API, die von den eigentlichen Bibliotheken, nämlich der Message Passing Library und der Master-Worker Library verwendet werden. Die Object Passing Library ermöglicht es, Objekte synchron oder asynchron an einen Empfänger per Unicast oder Broadcast zu versenden. Der Empfänger wird dabei über seine PeerID angesprochen.

2. Message Passing Library: Diese Bibliothek unterscheidet sich von der Object Passing Library lediglich darin, dass der Empfänger nicht über seine PeerID adressiert, sondern über einen nichtnegativen Integer Wert, dem sogenannten Rang. Der Rang eines Peers wird vom Controller - dessen einzige Aufgabe die Verwaltung und das Managment der Jobs und Peers zu übernehmen - berechnet und dem Peer mit einer Tabelle aller Ränge und den dazugehörigen PeerIDs die derzeit an dem Projekt beteiligt sind. Die Konsistenz der Tabelle der Ränge ist dabei Aufgabe des Controllers.

3. Master-Worker Library: Diese Bibliothek vereinfacht die Parallel-Programmier Ansätze, welcher auf einem Master-Worker Modell basieren. Hierbei schreibt der Programmierer zwei Programme, ein Programm für die Masterseite, welches die Rückgaben der Worker verarbeitet, und ein Programm für die Workerseite, welche mit den Daten arbeiten bzw. Berechnungen durchführen. Die Worker erhalten dabei eine Workunit, die eine bestimmte Anzahl von Berechnungen bzw Arbeit enthält, mit dem Worker-Code zusammen ergibt dies den eigentliche Job, welcher vom Master an den Worker übergeben wird und von diesem abgearbeitet wird.
   Die Planung und Vergabe der einzelnen Workunits ist dabei Aufgabe der Master-Worker Library, der Programmierer kann sich somit ganz auf das

eigentliche Problem konzentrieren.

[ShTS05]

2.4 Performance

Personal Power Plant ist bereits eine performante Anwendung, die jedoch weiter optimiert werden kann. So benötigt die TCP Implementierung in C lediglich 0.062 Milisekunden von der Übertragung bis zum Empfang von einem Byte, die TCP Implementierung von Java kommt an diesen Wert mit 0.064 Milisekunden schon sehr gut herran, jedoch benötigt das Message Parsing von Personal Power Plant für die Übertragung 4.5 Milisekunden und ist damit um den Faktor 72 langsamer als das reine TCP Protokoll von C.

Im Praxistest [ShTS05] mit einer Gigabit Ethernet Verbindung und 2,4 GHz

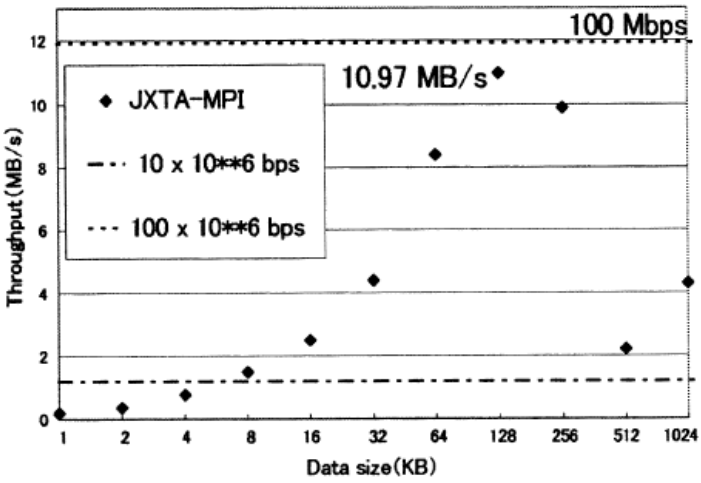


Abbildung 4. Netzwerkauslastung in P3

Xeon PC's hat sich ergeben, dass Personal Power Plant einen Spitzenwert von 10.97 MB/s erreicht wie aus Abbildung 4 ersichtlich. Damit bietet Personal Power Plant noch genügend Reserven um auch die nächsten Jahre die gänigsten Internet-Anbindungen komplett auszureizen, der Test zeigt jedoch auch, wie sehr die Netzwerkperformance noch verbessert werden kann.

Ein weitere wichtige Wert der für die Performance von Personal Power Plant Projekten erheblich ist, ist die Größe einer Workunit. Es zeigt sich, dass kleine Workunits größeren vorzuziehen sind. Der Performance Verlauf im Verhältniss zur Workunitgröße ist in Abbildung 5 genauer verdeutlicht. Je größer eine Workunit um so mehr Peers werden benötigt um wieder auf ein lineares Verhältniss zwischen der Größe der Workunit und der Abarbeitungsgeschwindigkeit zu kommen, was zum

einem am Voting liegt, siehe dazu Abschnitt 3.3.1, zum anderen jedoch an der Tatsache, das Knoten ausfallen, die Rechenleistung der Knoten stark schwankt, ...

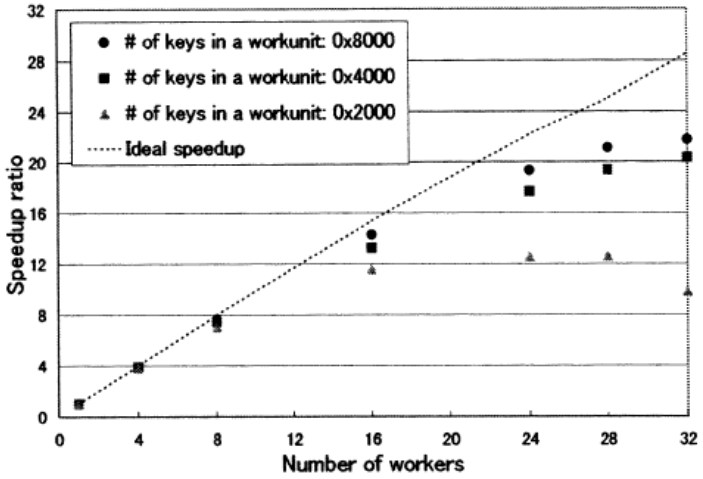


Abbildung 5. Performance in Abhänigkeit zur Workunitgröße

3 Gelöste Probleme von Peer-to-Peer Ansätzen des P3 Projektes

3.1 Firewall

Durch die Problematiken mit Würmen, Viren und Exploits, verstärken sowohl die Firmen als auch die Heimanwender ihre Sicherheitsmaßnahmen. Firewalls und Router gehören inzwischen zum Standard. Durch dieses wachsende Sicherheitsbedürfnis ist jedoch keine beliebige Netzwerkkommunikation zwischen zwei Rechnern mehr möglich. Doch genau diese beliebige Netzwerkkommunikation zwischen Rechnern wird für Peer-to-Peer Netze und damit auch für Personal Power Plant benötigt.

3.1.1 Lösung des Firewallproblems

Durch die Verwendung von JXTA , kann die Einschränkung durch Firewalls gelöst werden. JXTA liefert Firewall-Routing-Fähigkeiten, dabei verbindet sich der Host mit einem Peer ohne Einschränkungen, auch Relaypeer genannt und holt sich von diesem seine Nachrichten in periodischen Abständen ab. Dieser RelayPeer

ermöglicht eine Kommunikation über viel genutzte Standardports wie Port 80 (http), die in Firewalls meistens freigeschaltet sind.[Heis05]
Somit wird also ein Netzwerk-Overlay aufgebaut in dem alle Peers ohne Einschränkungen miteinander kommunizieren können, auch wenn sie im zugrunde liegenden Netzwerk durch eine Firewall oder NAT eingeschränkt sind. [ShTS05]

3.2 Hosts die unbeendete Jobs verlassen

Da ein Host einen Job jederzeit betreten als auch verlassen kann, was ohne Einschränkungen auch nach Beginn des Projektes möglich ist. Das Verlassen nach Erhalt einer Workunit kann mehrere Gründe haben:

- Der Host möchte das Projekt sabotieren.
- Der Host hatte einen Disconnect von seinem Internet-Service Provider.
- Der Host wurde in folge von Wartungsarbeiten heruntergefahren.
- ...

Deshalb muss garantiert werden, dass der Job durch dieses Phänomen nicht beeinträchtigt wird.

3.2.1 Lösungsansatz für AD-HOC-Disconnect

Die Master-Worker Library, welche die Workunits verwaltet, löst das Problem des Ad-Hoc-Verlassens einer Gruppe, in dem sie unbeendete Workunits einfach neu vergibt. Workunits werden auch neu vergeben, wenn ein Worker seine Aufgabe nach einer bestimmten Zeitspanne immer noch nicht abgearbeitet hat (Timeout).[ShTS05]
Damit wird als Nebeneffekt auf das Problem des so genannten Lazzy Workers mitgelöst. Als Lazzy Workers werden Knoten bezeichnet, welche nur Workunits empfangen, diese jedoch nicht bearbeiten.

3.3 Korrektheit der abgearbeiteten Workunits

Da es im Internet Parteien gibt, die Projekte aus den unterschiedlichsten Gründen sabotieren, muss ein Projekt, welches durch verteiles Rechnen gelöst werden soll diesen lästigen Aspekt berücksichtigen. Es kann also gut sein, dass Workunits zwar abgearbeitet werden, jedoch absichtlich falsche Ergebnisse zurückgeschickt werden um das Projekt zu sabotieren.

3.3.1 Lösung der Korrektheit duch Voting

Personal Power Plant verfolgt den Ansatz von Korrektheit durch Voting. Auch hier wird das Problem von der Master-Worker Library gelöst, der Programmierer braucht sich darum nicht zu kümmern.
Korrektheit durch Voting funktioniert folgendermaßen: Der Master verteilt eine Workunit m mal, dabei ist m ein im Projekt festgelegter Wert. Anschließend vergleicht er die m Rückgaben, eine Rückgabe gilt als korrekt, wenn sie mit j weiteren Rückgaben übereinstimmt.
In Personal Power Plant, müssen die Rückgaben exakt übereinstimmen, dies kann jedoch bei Fließkommazahlberechnungen problematisch werden, da unterschiedliche Hardware-Architekturen unterschiedlich runden. Aus diesem Grund ist es möglich die Prüffunktion der Master-Worker Library durch eine eigene Funktion zu ersetzten. [ShTS05]. Ein Problem tritt allerdings auf, wenn j bearbeitert Böse sind. Doch dieser Fall ist durch die zufällige Wahl von Peers für Abarbeitung eines Jobs meiner Ansicht nach eher unwahrsacheinlich.

4 Ungelöste Probleme des P3 Projektes

4.1 Problem des böswilligen Masters

Die Rolle des Masters, wird nicht zwingend vom Controller (Projekt-Ersteller) ausgeübt. Gibt es nur einen Bewerber für die Rolle des Masters, wählt der Controller diesen, gibt es keinen oder mehrere Bewerber, so wird der Master zufällig vom Controller ausgewählt. [ShTS05]
Da die Ergebnisse der Worker vom Master weiterverarbeitet werden, und nur die Rückgaben der Worker per Voting überprüft werden, kann somit ein Host, das Projekt sabotieren, sofern er dafür Sorge trägt, dass er die Rolle des Masters bekommt, in dem er z.B. alle anderen Hosts, die als Master kandidieren per Denial of Service Angriffes ausbremst oder lahmlegt um die Wahrscheinlichkeit für sich selbst zu erhöhen. Ist der Knoten Master kann er anschließend beliebig mit den Rückgaben verfahren. Auf Nachfrage bei einem der Entwicklern des Projektes Herrn Kazuyuki Shudo, habe ich erfahren, dass dieses Problem so nie berückstichtig worden ist, und deshalb keine Lösung dafür implementiert wurde.

4.1.1 Mögliche Lösungsansätze

Ich bin der Ansicht, das die Ausnutzen dieses Problem mit einfachen Mitteln, erschwert werden kann, einen Ansatz, den ich mir dafür überlegt habe:
Ein Ansatz ist, mit dem Master ähnlich wie mit den Worken zu verfahren. Man wählt nicht nur einen, sondern m Master zufällig über den Controller aus, der

Controller vergleicht nun die Rückgaben bzw. die Zwischenschritte der Master miteinander. Unterscheidet sich ein Master von i anderen, so würde man diesen ausschließen und einen anderen Host als Ersatz wählen.
Da der Controller, dem Rechner entspricht, welcher das Projekt eingestellt hat, kann man die Funktion des Controllers als korrekt ansehen.
Dieser Ansatz löst das Problem zwar nicht, senkt jedoch die Chancen eines Saboteurs, da er nun an i Master kommen muss.
Ich habe dieses Ansatz Herrn Kazuyuki Shudo vom Personal Power Plant Projekt vorgeschlagen, er ist meine Ansicht, dass diese Implementierung eine Möglichkeit wäre.

4.2 Disconnect des Masters

Ein weiteres Problem von Personal Power Plant ist die Eigenschaft, dass die Ergebnisse der Worker einzig und alleine dem Master mitgeteilt werden. Verlässt der Master nun die Gruppe z.B. durch:

- Einen 24h Disconnect, bedingt durch den Internet Service Provider.
- Einen Neustart des Rechners, bedingt durch Wartungsarbeiten, oder Softwarepatches.
- Stromausfall.
- ...

so sind alle bisher durchgeführten Berechnungen verloren. Stellt man sich nun ein Szenario vor, in dem fünf Jahre an einem Problem gerechnet wurde und anschließend der Master den Job verlässt, so wird die Tragweite und die daraus resultierende Gefahr deutlich, zumal die Wahl des Masters nicht beeinflusst werden kann. Es ist demnach theoretisch möglich, dass User zum Master kandidieren, welche über eine 24h Zwangstrennung verfügen.
Auch dieses Problem wurde auf Nachfrage bei Herrn Kazuyuki Shudo nicht berücksichtigt. Der Grund hierführ ist jedoch, das Japan das Problem der Zwangstrennung wohl nicht kennt, weshalb es ihnen nicht in den Sinn kamm, dass dieser Aspekt berücksichtigt werden muss.

4.2.1 Mögliche Lösungsansätze

Meiner Ansicht nach, kann auch dieses Problem gelöst werden, hierfür würden sich gleich zwei Verfahren anbieten, die ich mir überlegt habe:

1. Die Ergebnisse der Worker, werden in allen Peers der Projekte-Gruppe gespeichert. Fällt der Master aus, so werden die Ergebnisse durch Voting über den Controller an den neuen Master übergeben. Dies ist jedoch nur in einer sehr effizienten multicast Architektur realisierbar. Da der Controller immer der Knoten des Projekt-Gründers ist, gibt es keinen Grund der dagegenspricht.

2. Indem über mehrere stellvertretende Master gearbeitet wird. Der Master übergibt die Ergebnisse in periodischen Abständen an seine Stellvertreter, fällt der Master aus, übernimmt ein zufällig vom Controller gewählter Stellvertreter die Rolle des Masters und wählt einen neuen Stellvertreter für sich selbst, der seine alter Rolle als Stellvertreter einnimmt.

5 Fazit

Aus meiner Sicht, birgt die Verwendung von Peer-to-Peer Netzen für Verteiltes Rechnen ein großes Potential. Da durch diesen Ansatz, die Rechenleistung schon bei 30 Heimcomputer um das Zwanzigfache erhöht werden kann. [ShTS05]
Somit können viele Heimanwender, wenn die Resourcen gebündelt werden, große Projekte realisieren.
Zwar verursacht die Verwendung von Peer-to-Peer Netzwerken einige Probleme, doch wie in der Ausarbeitung gezeigt, existieren für die meisten Probleme sehr gute und effiziente Lösungen.
Allerdings bieten die Problematiken, welche sich durch Saboteure auf der Masterseite ergeben, als auch die Probleme durch das in Deutschland gut bekannte Phänomen der 24h Zwangstrennung, ein Feld für weiter Forschungsarbeiten. Denn niemand möchte sein Projekt, jeden Tag neu von vorne beginnen sehen.
Alles in allem ist Personal Power Plant eine Lösung, die dem Programmierer viele Aufgaben abnimmt, so dass mit wenig Aufwand interessante Projekte verwirklicht werden können, an die man sich sonst möglicherweise nicht heran getraut hätte, da der Aufwand zu groß gewesen wäre.

Literatur

[Heis05] Heiss, J. J.: “JXTA Technology Brings the Internet Back to Its Origin”. URL http://java.sun.com/developer/technicalArticles/JXTA/, Download vom 2.12.2006, Aug 2005. 8

[JXTA] “JXTA”. URL http://www.jxta.org/, Download vom 1.12.2006. 2

[Proj] “Liste der Projekte verteilten Rechnens”. URL http://de.wikipedia.org/wiki/Liste_der_Projekte_verteilten_Rechnens, Download vom 30.11.2006. 2

[Schi06] Schindelhauer, C.: “Peer to Peer Netzwerke”, Vorlesungsfolien: Einführung, Meilensteine, Überblick, 2006. 1, 2

[ShTS04] Shudo, K.; Tanaka, Y.; Sekiguchi, S.: “Makes over your PCs into power generator on the Grid”. URL http://www.shudo.net/publications/17th-APAN-P2P-and-Grid-200401/, 2004.

[ShTS05] Shudo, K.; Tanaka, Y.; Sekiguchi, S.: “P2P-based Middleware Enabling Transfer and Aggregation of Computational Resources”. URL http://www.shudo.net/publications/GP2PC2005/shudo-GP2PC2005-P3.pdf, May 2005. 2, 4, 6, 8, 9, 11