DG Network
Erweiterter P2P-Chat

Inhaltsverzeichnis

Einleitung

Was ist DG Network?

Mit DG Network können zwei Sachen gemeint sein: das von mir programmierte Chatprogramm oder das gleichnamige Netzwerk, mit dem das Programm arbeitet.
Letztes Jahr war DG Network (das Programm) fest mit dem Netzwerk verbunden, jetzt sind beide jedoch getrennt (auch wenn das Programm bisher nur über DG Network chatten kann und kein anderes Programm das Netzwerk unterstützt).
Das Programm DG Network ist ein Chatprogramm, mit dem sich die Benutzer über das Internet unterhalten können (normalerweise immer in „privaten“ Räumen für nur zwei Benutzer). Um im Internet mit meinen Freunden zu chatten, hatte ich bisher ICQ verwendet. ICQ hat zwar erweiterte Funktionen um sich z.B. Dateien zuzuschicken oder mit mehr als 2 Benutzern zusammen zu chatten, jedoch versagen diese Funktionen sobald einer der Teilnehmer einen Router zur Einwahl ins Internet verwendet.
Das Netzwerk DG Network ist ein von mir entworfenes Peer to Peer-Netzwerk. Ein Peer to Peer-Network ist ein dezentrales Netzwerk, in dem verschiedene, gleichberechtigte Rechner zusammengeschaltet sind und Anfragen weiterleiten. Da mein Netzwerk zum Chatten verwendet wird, unterstützt es das Routen von Paketen, also das Weiterleiten an ein bestimmtes Ziel (andere P2P-Netze unterstützen nur spezielle Funktionen, z.B. Suchen von Dateien). Im Gegensatz zum Internet (das ja auch ein P2P-Netzwerk ist), sind die Routing-Tabellen nicht statisch (eine Adresse hat eine feste Position innerhalb der Netzwerk-Topologie), sondern dynamisch. Dazu werden zwischen den Teilnehmern des Netzwerkes automatisch Routing-Informationen weitergegeben. Verschlüsselung und Bestätigung der Ankunft der Pakete gehört auch zu den Kernfunktionen meines Netzwerkes.

Warum noch einmal neu beginnen?

Ich habe das Programm DG Network bereits im vergangenen Jahr bei „Schüler experimentieren“ vorgestellt. Seitdem habe ich es fast vollständig verworfen und dafür neu begonnen. Dies ist nicht der erste Neubeginn bei der Programmierung von DG Network. Ich lerne immer mehr über P2P-Netze und Anwendungsprogrammierung mit .NET. Bei der Weiterprogrammierung von DG Network mussten immer wieder Abschnitte überarbeitet oder ganz durch neuen Code ersetzt werden. Dies habe ich immer gemacht, wenn mir die Architektur des alten Codes nicht (mehr) gefiel oder aber der Code nicht ohne große Veränderungen für ein benötigtes weiteres Feature angepasst werden konnte.
Das neue DG Network ist zum größten Teil neu, denn ich habe wieder von null angefangen und alles neu programmiert. Hätte ich nicht von null angefangen, hätte ich aber viele Teile auch auf den Stand geändert, auf dem sie jetzt stehen. Einige Teile des alten Codes gefielen mir aber bzw. es war nicht nötig, sie neu zu schrieben, also habe ich sie im neuen DG Network wieder verwendet. Aber auch bei den neu geschriebenen Komponenten habe ich immer wieder Codeabschnitte aus dem alten DG Network kopieren können.
Die eigentlichen Gründe für den Beginn von null war, dass die Plugin-Architektur im alten DG Network relativ schlecht war: ein Plugin war nur über Umwege dazu in der Lage, einen Menüeintrag in ein vorhandenes Menü einzufügen. Zum Beispiel musste ein Plugin folgendes durchführen, um das Kontextmenü der Freundesliste zu erweitern: Auf ein statisches Event bei Erzeugung des Chatformulars reagieren, dann bei diesem Formular einen Eventhandler für den Rechtsklick auf die Benutzerliste registrieren und dort das vorhandene Menü genau in dem Zeitpunkt zwischen Erstellung und Anzeige noch zu verändern.
Ein wichtiger Punkt für das Neuschreiben des Codes für das P2P-Netzwerkes war, dass das alte Netzwerk die Pakete abgesendet hat und keine Möglichkeit hatte, die Ankunft zu kontrollieren. Zwar konnte ein Antwortpaket zurückgeschickt werden, jedoch wurden die Routing-Tabellen nur alle 20 Sekunden weitergereicht. Ein Benutzer, der soeben erst online gegangen war, konnte zwar schon Pakete senden, es war für andere Benutzer aber nicht möglich, ihm zu antworten, bis sein Eintrag in den Routing-Tabellen auch bei ihnen angekommen war.
Und der wohl wichtigste Grund für das Neuschreiben der gesamten Anwendung war, dass es DG Network möglich sein sollte, sich mit mehreren Netzwerken gleichzeitig zu verbinden.

Das alte DG Network war fest mit dem „einen DG Network“ verdrahtet, das neue hingegen kann per Plugin um weitere Netzwerke erweitert werden, selbst wenn diese ganz anders aufgebaut sind. Ein Beispiel dafür ist das IRC-Plugin, mit dem der Chat auf einem IRC-Server möglich ist. Des Weiteren ist ein Plugin für ICQ geplant.

DG Network - das Netzwerk

Das P2P-Netzwerk habe ich bereits letztes Jahr erklärt, es hat sich nur relativ wenig geändert.
Die Änderungen sind für den Benutzer des Chatprogramms nicht sichtbar (Normalbenutzer bemerken kaum, dass ein P2P-Netzwerk verwendet wird), aber für die Struktur von DG Network macht es viel aus.
Während im alten DG Network der Netzwerk-Code mit dem Chat-Code bunt gemischt war, ist sämtlicher Netzwerkcode nun in einer eigenen .dll-Datei, der Peer to Peer library, ausgegliedert. Somit könnten auch andere Programme mit der Bibliothek eigene P2P-Netze aufbauen.

Wie zuvor bauen hier alle am Netzwerk teilnehmenden Computer ein Netzwerk innerhalb des Internets (und in lokalen Netzwerken), über das Pakete an das Ziel weitergeleitet werden. Das eigene Routing ist nötig, da nicht jeder Benutzer eingehende Verbindungen aus dem Internet annehmen kann – Router und böse Netzwerkadministratoren könnten dies verhindern. Ausgehende Verbindungen aufzubauen ist allerdings immer erlaubt (sonst könnte man nicht mal surfen), deshalb baut DG Network ausgehende Verbindungen zu anderen Benutzern auf und bekommt dann von diesen die Pakete weitergeleitet.
Das Problem beim Weiterleiten ist, das zwar jeder Teilnehmer am Netzwerk die Benutzer kennt, mit denen er verbunden ist, er kennt aber nicht den Aufbau des gesamten Netzwerkes. Damit die Pakete an das richtige Ziel weitergeleitet werden können, benötigt DG Network Routing-Tabellen. Diese werden automatisch aufgebaut und durch Kommunikation mit den Nachbar-Benutzern im Netzwerk aktualisiert und weitergereicht.
In den Routing-Tabellen steht im Prinzip, welcher Benutzer über welche Route am besten zu erreichen ist. Es ist aber nicht die ganze Route bekannt, sondern jeweils nur die nächste Station.
Um die Tabellen abzugleichen, sendet jeder Benutzer alle 10 Sekunden seine Tabelle an seine Nachbarn (um Bandbreite zu sparen, werden nur die Änderungen gesendet). DG Network verbindet beim Empfang einer Tabelle die empfangene Routing-Tabelle mit der eigenen und behält somit nur den kürzesten Weg (teilweise werden aber auch alternative Wege mitgespeichert, aber nur der kürzeste Weg wird weitergegeben). Ein weiteres Feld in den Routing-Tabellen wird benutzt, um die Einträge der beendeten Verbindungen zu löschen. Dazu enthält jeder Eintrag in den Routing-Tabellen nicht nur die Nummer des Benutzers, für den der Eintrag steht, sondern auch die Nummer des Benutzers, mit dem er verbunden ist (die Verbindung, durch die der Eintrag erzeugt wurde). Wenn nun diese Verbindung geschlossen wird, können alle Routen, die auf dieser Verbindung basieren, wieder aus den Tabellen entfernt werden und durch die vorher gespeicherten Alternativ-Routen ersetzt werden.

Beim Senden der Pakete hat sich im Gegensatz zum alten DG Network verändert, dass sich jedes Paket nun den Weg „merkt“, über das es gekommen ist. Zusätzlich hat jedes Paket eine „Antwort-ID“ im verschlüsselten Teil. Mit dieser ID kann der Empfänger des Paketes den Empfang bestätigen. Dazu sendet er ein Antwort-Paket zurück, die Antwort-ID garantiert, dass nur der echte Empfänger (der das Paket entschlüsseln konnte) die Bestätigung senden kann. Antwort-Pakete sind keine normalen Pakete, denn sie werden nicht anhand der Routing-Tabellen gesendet, sondern gehen den Weg des Paketes, das sie bestätigen sollen, rückwärts. Dadurch kann auch ein Benutzer, dessen Eintrag in den Routing-Tabellen noch nicht durch das Netzwerk verteilt wurde, bereits Antworten auf seine Pakete empfangen. Schlägt das Senden auf der Rückroute fehl, verpackt DG Network die Antwort in einem normalen Paket und verschickt dies mit der üblichen Methode direkt ans Ziel.

Die größte Neuerung hat bei der Verschlüsselung stattgefunden: während vorher nur bestimmte Pakete verschlüsselt wurden (nämlich der Chat innerhalb von Gruppen), kann jetzt alles direkt auf Paketebene verschlüsselt werden.
Dazu tauschen die Kommunikationspartner zunächst ihre öffentlichen Schlüssel aus und einigen sich über die asymmetrische Verschlüsselung auf einen synchronen Schlüssel für die eigentlichen Daten.

Diese Technik schützt zwar vor bloßen Zuhörern, ist aber gegen Man-in-the-middle-Attacken nicht geschützt. Und da alle Pakete über verschiedene Benutzer im Netzwerk weitergeleitet werden, ist eine solche Attacke wesentlich einfacher als bei direkten TCP/IP-Verbindungen. Wie auch bei anderen Anwendungen der asymmetrischen Verschlüsselung zum Schlüsselaustausch kann dies durch Zertifikate verhindert werden. Dazu müsste z.B. der Einwahlserver ins Netzwerk (dessen öffentlicher Schlüssel dann allgemein bekannt wäre) bei der Einwahl jedem Benutzer ein Zertifikat ausstellen, das seinen öffentlichen Schlüssel seiner Benutzer-ID zuordnet. So könnte der andere beim Schlüsselaustausch überprüfen, ob der öffentliche Schlüssel nicht durch eine Man-in-the-middle-Attacke gefälscht wurde.
Dazu müsste allerdings der Einwahlserver vertrauenswürdig sein und das Ausstellen von Zertifikaten unterstützen. Der IP-Server ist im Moment in PHP programmiert, da mein Provider nichts anderes unterstützt, und für PHP ist mir keine Bibliothek mit Verschlüsslungsfunktionen bekannt.

Die Plugin-Architektur

Anforderungen

Für die Plugin-Architektur gab es hohe Anforderungen: es sollte möglich sein, Menüs zu erweitern, ohne erst auf irgendwelche Ereignisse reagieren zu müssen.
Die Plugin-Architektur sollte auch die Ressourcen-Dateien verwalten, denn das neue DG Network sollte auch übersetzbar sein (bisher sind nur deutsch oder englisch als Sprachen auswählbar).
Plugins sollten auch in der Lage sein, neue Chat-Netzwerke im Programm zu integrieren. Plugins sollten auch Teile der Benutzeroberfläche und der direkt damit verbundenen Funktionalität ersetzen bzw. verändern können, oder aber dem Benutzer die Wahl zwischen den verfügbaren Komponenten für die Benutzeroberfläche geben.
Das Anzeigefenster für die empfangenen Chatmitteilungen ist z.B. standardmäßig nur eine simple Textbox, die die empfangenen Mitteilungen in Textform (also ohne Formatierungen oder Bildern) anzeigt. Mit DG Network werden aber zwei Plugins installiert, die dem Benutzer die Möglichkeit bieten, zwei weitere „Displays“ für die empfangenen Mitteilungen zu wählen: Ein Plugin benutzt den Internet Explorer, um die Mitteilungen darzustellen (wie schon letztes Jahr), das andere Plugin nutzt eine ähnliche Methode für Mozilla.
Allein dadurch, dass die Plugins sich im Plugin-Ordner befinden, sollte dem Benutzer die Möglichkeit gegeben werden, zwischen den drei möglichen Displays zu wählen.

Außerdem sollten Plugins die Möglichkeit haben, selbst von anderen Plugins erweitert zu werden. Menüs, die z.B. durch das (zukünftige) ICQ-Plugin definiert werden, sollten von anderen Plugins so einfach erweitert werden können, als wären es Menüs von Standardfunktionen in DG Network.

Die Lösung

Mehr durch Zufall fand ich die Lösung für alle diese Anforderungen bereits fertig implementiert in einem anderen Programm. Als die Open Source-Entwicklungsumgebung „SharpDevelop“ (www.sharpdevelop.net) ins Beta-Stadium kam, habe ich mich stärker dafür interessiert und schließlich in den Sommerferien angefangen, bei der Programmierung von SharpDevelop mitzuhelfen. Das war gerade, als ich mich entschlossen hatte, DG Network neu zu beginnen und eine passende Plugin-Architektur suchte. SharpDevelop bietet eine Architektur, die genau diese Anforderungen erfüllt. Und da SharpDevelop Open Source ist, konnte ich sie problemlos in DG Network wieder verwenden. Der Kern der Architektur steckt in einer Datei namens ICSharpCode.Core.dll und ist vollständig von dem Programm, das ihn verwendet, unabhängig. Für DG Network musste ich nur sehr wenige Änderungen am Programmcode des Core durchführen.

Die Komponenten

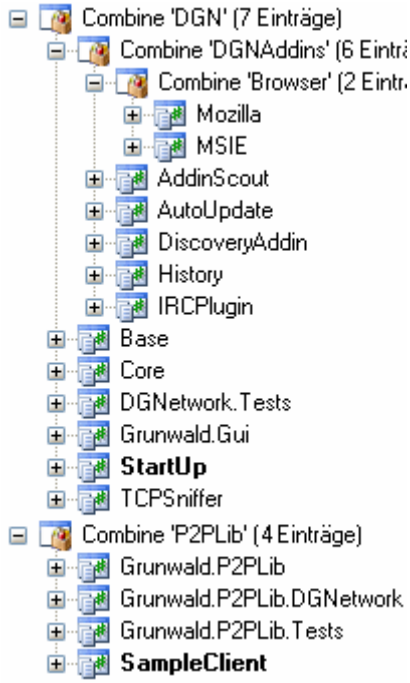

Abbildung 1: Die Komponenten

Jedes der im Bild links aufgeführten Einträge ist ein Projekt und wird in .dll oder .exe-Datei kompiliert.
Mozilla bzw. MSIE sind die beiden Browser-Addins für die Darstellung der empfangenen Mitteilungen.
Der AddinScout (aus SharpDevelop übernommen) ist ein Addin, mit dem der AddIn-Tree (siehe nächste Seite) angezeigt werden kann. AutoUpdate ist ein AddIn, mit dem neue DG Network-Versionen automatisch von meiner Homepage heruntergeladen werden. Das DiscoveryAddin übernimmt die Dienste für das Netzwerk, die einen Server (eine php-Seite auf meiner Homepage) benötigen: IP-Server und Suche nach Benutzern.
History ist ein kleines Addin, dass alle empfangenen und gesendeten Mitteilungen im HTML-Format abspeichern kann.
IRCPlugin fügt die Unterstützung für IRC hinzu.
Base ist das größte Projekt, es enthält sämtliche Basis-Funktionalität, also die Benutzeroberfläche, die Klassen für den Chat, Basisklassen für Suche usw.
Core ist aus SharpDevelop übernommen und ist der Kern des Plugin-Systems.
DGNetwork.Tests enthält Unit-Tests zum automatischen Testen der Funktionen in Base.
Grunwald.Gui ist meine eigene Bibliothek, die Klassen für die benötigten Controls für die Benutzeroberfläche enthält. StartUp ist das Startprojekt, das zu der Datei „DG Network.exe“ kompiliert wird. TCPSniffer ist ein von mir programmiertes Tool, um TCP-Verbindungen zu überwachen (nützlich für die Fehlersuche in DG Network). Die P2PLib stellt die Bibliothek dar, um sich mit dem Netzwerk DG Network zu verbinden. Sie besteht aus P2PLib.dll, die für viele P2P-Netze wieder verwendbare Klassen enthält (zum Beispiel einen Webserver), und P2PLib.DGNetwork.dll, die das DG Network Protokoll implementiert. Dazu kommt noch ein Projekt mit automatischen Unit-Tests für P2PLib und ein Test-Client, mit dem die Funktion der Routing-Tabellen getestet und veranschaulicht werden kann.

AddIn-Tree

Der AddIn-Tree ist die Basis der gesamten Plugin-Architektur. Er hält sozusagen die verschiedenen Komponenten zusammen. Auch wenn das Prinzip des AddIn-Trees nicht von mir stammt, muss ich es erklären, damit die Struktur von DG Network verstanden werden kann.
Man kann sich den AddIn-Tree als eine Art Dateisystem vorstellen, denn er ist auch hierarchisch strukturiert und jeder Knoten erhält einen Namen nach seinem Pfad.
Definiert wird der AddIn-Tree per XML. Der SharpDevelop Core lädt die XML-Dateien aller Plugins (jedes Plugin hat seine eigene XML-Definition) und führt diese zusammen.
Hier ein Beispiel aus DG Network:

```
<Extension path = "/TNAIcon">
      <TNAIcon id      = "MainIcon"
               tooltip = "DG Network"
               icon    = "Icons.App">
            <MenuItem id    = "ShowMainWindow"
                      label = "${res:Menu.ShowMain}"
                      icon  = "Icons.DGNetwork"
                      class = "DGNetwork.Commands.MenuShow"/>
            <MenuItem id    = "Status"
                      label = "${res:Menu.Status}">
                  <MenuItem id    = "Online"
                            label = "${res:Menu.Status.Online}"
                            icon  = "Icons.Status.Online"
                            class = "DGNetwork.Commands.SetStatus"/>
                  ....
                  <MenuItem id    = "Offline"
```

```
                        label = "${res:Menu.Status.Offline}"
                        icon  = "Icons.Status.Offline"
                        class = "DGNetwork.Commands.SetStatus"/>
            </MenuItem>
            <MenuItem id = "Separator1" label = "-"/>
            <MenuItem id    = "Monitor"
                      label = "${res:Menu.Monitor}"
                      icon  = "Icons.Monitor"
                      class = "DGNetwork.Commands.MenuMonitor"/>
            ....
            <MenuItem id = "Separator2" label = "-"/>
            ....
      </TNAIcon>
</Extension>
```

Dies ist die Definition des Icons in der Startleiste mit dem entsprechenden Kontextmenü.
`<Extension path = "/TNAIcon">` gibt an, dass neue Items in den AddIn-Tree in de Ordner /TNAIcon eingefügt werden. /TNAIcon ist ein Pfad in dem AddIn-Tree. Darin wird der Knoten MainIcon erzeugt. MainIcon hat damit die „Adresse“ /TNAIcon/MainIcon.
MainIcon ist aber kein „Ordner“ im AddIn-Tree, sondern hat ein Codon angefügt. Dieses Codon ist vom Typ TNAIconCodon (eine von mir programmierte Klasse in Base.dll). Es hat Menüeinträge als Unterelemente, die die Menüeinträge des Kontextmenüs darstellen. Der Menüeintrag Online hat somit die Adresse /TNAIcon/MainIcon/Status/Online.
Jeder Menüeintrag hat ein Icon, ein Label (Name der Beschriftung für den Eintrag, als Name der Ressource in den Übersetzungsdateien) und eine Klasse angefügt. Die Klasse gibt den Namen (mit vollständigem Namespace) einer Klasse an, die die Schnittstelle IMenuCommand implementiert. Die Methode Run() einer solchen Klasse wird ausgeführt, wenn der Benutzer auf den Menüeintrag klickt.
Wie kann ein Plugin nun dieses Menü erweitern? Ganz einfach, jedes Plugin hat eine eigene XML-Datei, die ein Stück des AddIn-Trees definiert.
Um z.B. einen weiteren Status-Eintrag hinzuzufügen, muss nur folgender Code eingebunden werden:

```
<Extension path = "/TNAIcon/MainIcon/Status">
      <MenuItem id    = "DND"
                label = "${res:Menu.Status.DND}"
                icon  = "Icons.Status.DND"
                class = "DGNetwork.Commands.SetStatus"
                insertafter  = "Out"
                insertbefore = "Invisible"/>
</Extension>
```

Ganz gleich in welcher Reihenfolge die Plugins geladen werden (der Core sieht Base.dll auch als Plugin an), es wird immer der gleiche AddIn-Tree erzeugt.
Mit den Informationen in diesem durch den Core vorbereiteten AddIn-Tree kann DG Network dann die Menüs erzeugen. Dies ist die einzige wichtige Funktionalität, die DG Network aus dem SharpDevelop Core nutzt. Durch die korrekte Verwendung des AddIn-Trees werden alle unter „Anforderungen“ beschriebenen Probleme gelöst.
Wie an der Label-Eigenschaft der MenuItems zu sehen, ist DG Network übersetzbar. Sämtliche für den Benutzer sichtbaren Texte sind in einer Ressourcendatei gespeichert und können dort über einen Key (z.B. Menu.Status.DND) abgerufen werden. Die Ressourcendatei, die verwendet wird, ist abhängig von der eingestellten Sprache. Auf Menu.Status.DND kann so „Do not disturb“ oder „Bitt nicht stören“ werden.

DG Network – das Chatprogramm

Suche nach Benutzern

Die Suche in DG Network ist ein relativ großes Problem: es können mehrere Netzwerke (DGN, ICQ, …) durchsucht werden und jedes Netzwerk hat einen eigenen Satz Suchkriterien. Der Benutzer sollte aber in der Lage sein, in mehreren Netzwerken gleichzeitig zu suchen.

Im Netzwerk DG Network selbst gibt es bei der Suche das Problem, dass in einem dezentralen Netzwerk eine gute Suchfunktion nur schwer zu programmieren ist, da es (im Gegensatz zu Suchen in File-Sharing-Netzen) nötig ist, **alle** Suchergebnisse zu finden und nicht nur einige.
Das Suchsystem in DG Network muss also generell erweiterbar sein. Die Suche für das Netzwerk sucht im Moment über den Einwahlserver (danielgrunwald.de), wo sich alle Benutzer registrieren können, es ist aber möglich, dass dies später durch eine P2P-Suche ersetzt wird.

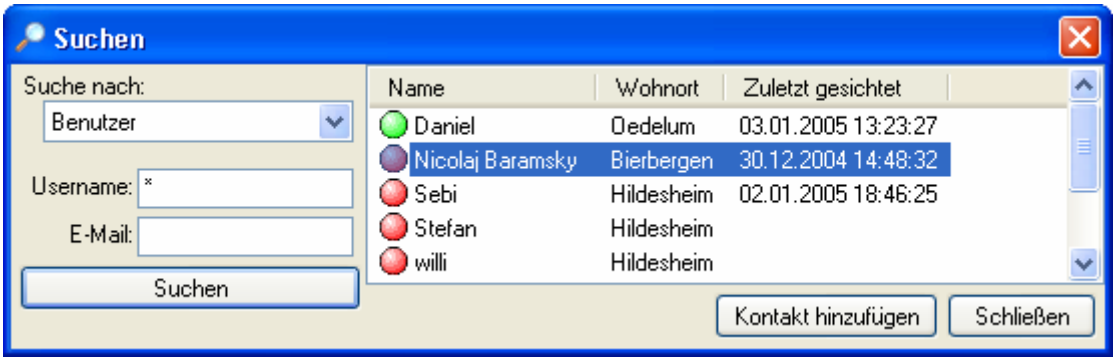

Abbildung 2: Der Suchdialog

Hier ein (vereinfachtes) Diagramm vom DGNetwork.Search Namespace:

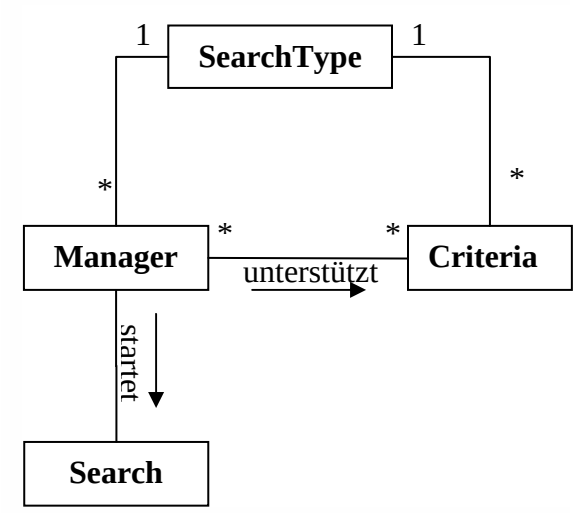

Abbildung 3: Diagramm der Suche

SearchType ist der Typ der Suche, also Suche nach Benutzer oder Suche nach Gruppe. Diesen kann der Benutzer in der Combobox wählen (siehe Abbildung 1).
Criteria ist ein Suchkriterium, das der Benutzer in Form einer Textbox eingeben kann. Der Manager kann eine Suche starten. Dabei wird ein Objekt der Search-Klasse zurückgegeben, das eine Liste der Suchergebnisse enthält. SearchType, Manager und Criteria werden über den AddIn-Tree definiert, so können Plugins also neue Manager und Criteria hinzufügen.
Im Moment gibt es nur eine Suche nach Benutzern, die im Moment auch nur einen einzigen Suchmanager unterstützt. Dieser greift auf eine PHP-Seite auf meiner Homepage zu und lädt von dort die Suchergebnisse herunter.
Um dort gefunden zu werden, kann sich jeder Benutzer beim ersten Start von DG Network (oder später in den Einstellungen) einen Account in der Datenbank auf meiner Homepage anlegen. Dort muss er Benutzername und Passwort wählen und kann dazu noch weitere Informationen wie Homepage oder Wohnort angeben, die dann in der Suchergebnisseite angezeigt werden.

Kontaktliste

Abbildung 4: Kontaktliste

Auch die Kontaktliste auf die Unterstützung von mehreren Netzwerken vorbereitet sein. Dies gestaltet sich relativ einfach, ein Eintrag für die Kontaktliste muss eine Schnittstelle implementieren, die folgende Funktionen definiert: Name und Symbol abrufen, Doppelklick ausführen, Kontextmenü abrufen, Kopie anlegen und das Element in XML umwandeln und zurückwandeln (um die Kontaktliste abzuspeichern).
Eine andere Schnittstelle erweitert den Satz der Funktionen um das Abrufen und Ändern der Unterelemente. Kontaktlisteneinträge, die auch diese Schnittstelle implementieren, können als Ordner benutzt werden, um andere Einträge einzuordnen.
Um die Kontaktliste anzuzeigen, habe ich ein eigenes TreeView-Control programmiert, das von dem normalen TreeView im .NET Framework erbt. Hinzugefügt habe ich die Funktion, dass man per Drag'n'Drop die Kontakte verschieben kann. Man kann entweder nur die Reihenfolge ändern oder aber Element in oder aus Ordner schieben, oder aber Elemente oder ganze Ordner durch drücken von Strg kopieren.
Im Hauptmodul von DG Network gibt es nur einfache Gruppen zum Sortieren von anderen Kontakten und DG Network Kontakte (also Freunde, die das Netzwerk DG Network benutzen).
Jeder der Einträge kann ein eigenes Kontextmenü definieren, indem er den Pfad angibt, in dem im AddIn-Tree das Kontextmenü gespeichert ist. Auch die leere Fläche hat ein Kontextmenü mit der Option „neue Gruppe" zugeordnet, und alle Kontextmenüs können mit dem AddIn-Tree erweitert werden. Das IRC-AddIn erweitert so das Menü von Gruppen um die Option „IRC-Server" hinzufügen, was einen Eintrag für einen IRC-Server auf der Kontaktliste erzeugt.

Benutzerstatus in der Kontaktliste

Die Kontaktliste zeigt immer den aktuellen Status des Benutzers an. Dies passiert durch das Statussystem. Zunächst kann jeder Benutzer über das Menü auf dem Startleisten-Icon seinen Status wählen.

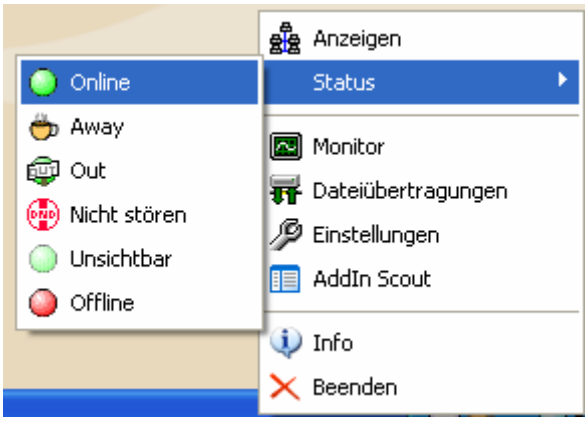

Abbildung 5: Wahl des Status

Wenn ein Benutzer auf Online gestellt ist, aber länger als zwei Minuten nicht mehr die Maus bewegt oder die Tastatur benutzt, wechselt DG Network automatisch auf Away, nach zehn Minuten dann auf Out. Sobald der Benutzer wieder an seinen PC zurückkehrt, wechselt DG Network dann automatisch auf Online zurück. Status-Änderungen müssen über das P2P-Netzwerk an alle Benutzer geschickt werden, die uns auf ihrer Kontaktliste haben. Dies entspricht nicht unbedingt den Benutzern, die wir auf unserer Liste haben.
Deshalb schickt DG Network alle zwei Minuten ein Paket an alle Benutzer auf unserer Freundesliste. Dieses Paket enthält den eigenen Status und die Anfrage „ich bin an deinem Status interessiert". Wenn eine Anfrage nach dem eigenen Status eintrifft, wird der Absender auf einer Liste vermerkt. An diese

Liste wird immer dann ein Paket gesendet, wenn sich der eigene Status ändert. Indem ein Paket ohne das Interesse-Flag gesendet wird, kann ein Benutzer sich auch wieder aus der Liste entfernen. Dies macht DG Network z.B. automatisch, wenn ein Kontakt von der Liste gelöscht wird.
Durch das Anlegen einer Liste der interessierten Benutzer wird der Status wenn nötig immer sofort bei Änderungen übertragen. Durch das Mitsenden des Interesse-Flags kann sich das System von Fehlerzuständen erholen (wenn z.B. die Verbindung zusammenbricht und das Austragen nicht mehr möglich ist). Zusätzlich werden alle Einträge von der Liste gelöscht, von denen innerhalb von 5 Minuten kein Status-Paket mehr empfangen wurde (deshalb das Senden alle 2 Minuten, auch wenn sich nichts geändert hat). Status-Pakete nutzen **nicht** das integrierte Paket-Sicherungssystem und Antwortsystem, indem die Paket-ID und Antwort-ID auf 0 gesetzt wird, denn eine Bestätigung dieser Pakete wäre überflüssig (sie werden durch die alle 2 Minuten erneuten Anfragen „verzögert“ bestätigt) und würde die Datenübertragung im Netzwerk stark steigern.

Der Chat

Auch der Chat sollte so erweiterbar wie möglich sein. Ein gemeinsames Chatfenster soll den Chat in verschiedenen Netzwerken ermöglichen. Und auch die Benutzeroberfläche soll durch Plugins erweiterbar sein.

Dynamische Benutzeroberfläche

Während Menüs sich leicht erweitern lassen, indem ein Eintrag hinzugefügt wird, ist nicht besonders klar, was man sich unter einer generell erweiterbaren Benutzeroberfläche vorstellen soll. Das gesamte Formular aus Definitionen aus dem AddIn-Tree zu erstellen wäre zwar möglich, ist aber relativ aufwändig und bringt nicht das gewünschte Ergebnis, denn die Funktionalität hinter den Elementen lässt sich nicht direkt als xml darstellen.
Außer den Menüs und der Toolbar, die ja bereits per AddIn-Tree erweiterbar sind, gibt es nur zwei Controls, deren Erweiterung bzw. Austausch sinnvoll erscheint: das Eingabefeld (genannt EditBox) und die Anzeige der empfangenen Chatnachrichten (Display).
Bei diesen beiden hat der Benutzer in den Optionen die Möglichkeit, eine Implementation zu wählen. Ein Plugin kann (natürlich per Definition im AddIn-Tree) eine weitere Option in der Einstellungsseite hinzufügen. Für die EditBox kann zwischen einem einfachen Texteingabefeld und einer RichTextBox, die auch Formatierungen unterstützt, gewählt werden. Beim Display besteht die Wahl zwischen dem Internet Explorer oder dem Mozilla Control (ein ActiveX-Control, um den Mozilla Browser in Windows Anwendungen zu verwenden).

Abbildung 6: Das Chatfenster

Erweiterbares Hauptfenster

Die Konstruktion des Hauptfensters ist auch über den AddIn-Tree erweiterbar.
Während das normale Hauptfenster nur die Kontaktliste enthielt, ist jetzt auch eine kombinierte Ansicht Chatfenster / Kontaktliste verfügbar. So kommt DG Network mit einem Fenster aus, was die Benutzerfreundlichkeit stark erhöht, da der Chatpartner gewechselt werden kann, ohne dass dafür ein neues Fenster geöffnet werden muss.
Welcher Hauptfenstertyp verwendet werden soll, kann in den Einstellungen gewählt werden.

Abbildung 7: Hauptfenster mit Kontaktliste und Chatfenster

Die Erstellung des Hauptfensters verhält sich ähnlich wie die von Display und EditBox: Die Liste der verfügbaren Fenstertypen wird aus dem AddIn-Tree generiert und ist somit erweiterbar. Der Benutzer kann dann einen Typ in den Einstellungen wählen, und die dazu zugeordnete Klasse wird dann verwendet, um das Fenster aufzubauen.

Mitteilungen senden

Beim Drücken des Sende-Buttons wird die EditBox angewiesen, ihren aktuellen Inhalt an den aktiven Chatraum zu senden. Dazu wird zunächst ein Objekt vom Typ Message erzeugt. Dieses nimmt den Inhalt der Box in drei Formaten auf: HTML, RTF und reiner Text. Die EditBox setzt dazu die Formate, die ihr bekannt sind (d.h. die einfache Textbox setzt den reinen Text und die RichTextBox den RTF-Code). Dann wird das Message-Objekt an den Chatraum übergeben. Der Chatraum ist ein Objekt, dessen Klasse die Schnittstelle IChatRoom implementiert. Ein Chatraum hat Methoden um Mitteilungen zu senden, zu empfangen und die Liste der Benutzer im Chatraum abzurufen. Beim normalen Chat sind alle Chaträume Instanzen der Klasse DGNPrivateChat, d.h. man chattet nur mit einem anderen Benutzer über das DG Network Protokoll. Plugins könnten beim Doppelklick auf einen ihrer Einträge in der Kontaktliste den Chat natürlich mit einem anderen Chatraum öffnen, der die Pakete dann in dem für sie passenden Format sendet. Somit ist der Chatraum dafür zuständig, die Mitteilung in ein Datenpaket für das entsprechende Netzwerk zu verpacken. Doch bevor dies geschieht, schickt der Chatraum die Mitteilung durch eine Kette von Filtern.
Die Liste der Filter ist im AddIn-Tree definiert. Im Moment ist nur der HtmlFilter in Benutzung. Dieser lädt aus der Mitteilung den HTML-Code (die Klasse Message führt Konvertierungen zwischen den drei Formaten wenn nötig automatisch aus) und versucht, alle HTML-Tags zu parsen. Auch Tags im Stil von Forencode ([b] anstatt von <b>) wird geparst und in den entsprechenden HTML-Code konvertiert.
Per AddIn-Tree können Plugins bei bestimmten HTML-Tags bestimmte Aktionen definieren. Dies nutzt z.B. das Dateiübertragungsmodul um an Stelle von <sendfile>-Tags die gesendete Datei

einzufügen. Auch Text-Ersetzungen können an dieser Stelle durchgeführt werden. So können Smilies durch Links zu den entsprechenden Bildern verschickt werden.
Die fertig gefilterte Mitteilung wird schließlich netzwerkabhängig versandt.

Mitteilungen empfangen

Wird eine Mitteilung über ein Netzwerk empfangen, muss sich das entsprechende Netzwerk darum kümmern, den entsprechenden Chatraum zu finden bzw. anzulegen. Der Chatraum kann sich dann um das weitere Empfangen des Paketes kümmern. Zunächst wird das Paket wieder in ein Objekt vom Typ Message umgewandelt. Dann muss sichergestellt werden, dass der Benutzer die Mitteilung auch sehen kann. Ist kein Fenster mit dem Chatraum geöffnet, so wird die Mitteilung gespeichert, bis der Benutzer das Fenster öffnet. Außerdem wird eine Benachrichtigung angezeigt, dass eine Mitteilung bereit liegt. Durch das Klicken auf die Benachrichtigung gelangt der Benutzer direkt in den richtigen Chatraum.

Ist bereits ein Fenster für den Chatraum geöffnet, wird die Mitteilung dort angezeigt. Sollte das Fenster offen, aber nicht das aktive Fenster sein, so wird die Windows-Funktion Flash() verwendet, um den Button des Fensters auf der Taskleiste blinken zu lassen.
Das normale Paket-Antwort-System das DG Network-Protokolls wird verwendet, um den Empfang der Mitteilung zu bestätigen.

Mitteilungen anzeigen

Beim Empfänger wird die Mitteilung bei offenem Fenster beim Empfang der Mitteilung angezeigt; beim Absender erst, wenn der Empfang der Mitteilung bestätigt wurde (durch das Paket-Antwortsystem).
Bei beiden Rechnern läuft die Mitteilung nun wieder durch eine Kette von Filtern – diesmal die Empfangsfilter. Auch hier ist ein HtmlFilter bisher der einzige Filter. Er hat eine noch wichtigere Aufgabe als der Filter beim Senden, denn er schränkt den HTML-Code, der verwendet werden kann, stark ein. Dazu wird der HTML-Code wieder geparst, diesmal allerdings mit strengeren Regeln. Bei Fehlern in einem Tag wird der Tag als ungültig markiert und das Startzeichen < in < umgewandelt. Bei Tags mit korrekter Syntax wird dann der Inhalt der Tags geprüft. Nur Tags, deren Name im AddIn-Tree registriert ist kommen durch, alle anderen werden aussortiert. Somit werden gefährliche Tags wie <script> aussortiert. Auch bei den Attributen kommt nur eine kleine Anzahl erlaubter Attribute durch, <img src="…"> ist erlaubt, aber <img onMouseOver="alert('böses Javascript');"> ist verboten.
Auch sorgt der Filter dafür, dass Tags wie <b> korrekt geschlossen werden, damit die nachfolgenden Chateinträge nicht alle fett erscheinen.
Auch hier können per AddIn-Tree weitere Tags eingefügt werden. Ein spezieller Tag der Dateiübertragung sorgt zum Beispiel für die Anzeige des speziellen Links zum Download der Datei.

Mehrbenutzerchat

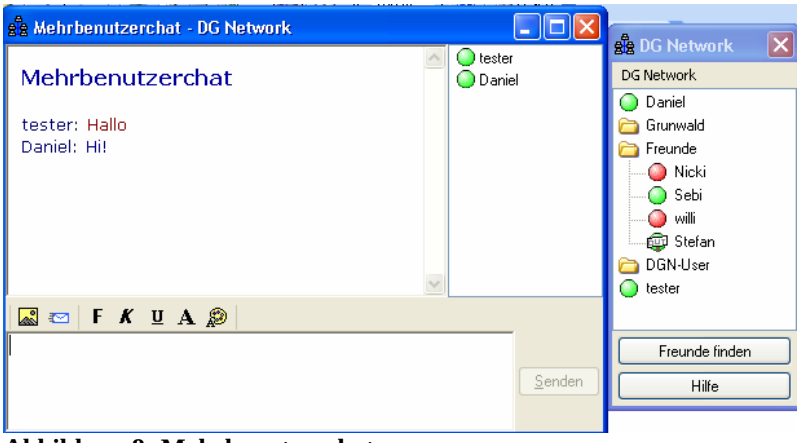

Abbildung 8: Mehrbenutzerchat

Der Mehrbenutzerchat bietet die Möglichkeit, über das DG Network Peer to Peer-Netzwerk mit mehreren anderen Benutzern auf einmal zu chatten. Mit einem Klick auf „Mehrbenutzerchat starten“ im Kontextmenü eines Benutzers auf der Kontaktliste kann ein solcher Chatraum erzeugt werden. Er erscheint im neuen Fenster mit einer Teilnehmerliste am rechten Rand. Per Drag’n’Drop können von der Kontaktliste weitere Benutzer in den Mehrbenutzerchat gezogen werden, diese werden dann auch in den Chat eingeladen. Um einen Mehrbenutzerchat wieder zu verlassen reicht es aus, das entsprechende Fenster zu schließen.
Der Mehrbenutzerchat benutzt intern Multicast-Pakete (eine Funktion des Peer to Peer-Netzwerkes), um die Pakete an alle bekannten Teilnehmer zu schicken. Durch das im Peer to Peer-Netzwerk integrierte Antwortsystem werden negativ oder nicht antwortende Benutzer aus der Benutzerliste des Mehrbenutzerchats entfernt. Außerdem wird beim Senden von Paketen jeweils die komplette Benutzerliste mitgeschickt, so dass die Liste sich automatisch zwischen allen Benutzers synchronisiert.
Alle Pakete im Mehrbenutzerchat werden mit dem symmetrischen Rijndael-Verfahren (Advanced Encryption Standard) verschlüsselt. Der SHA1-Hash des Schlüssels wird als eine Art ID für den Mehrbenutzerchat verwendet. Der Schlüssel selbst wird zusammen mit der Einladung an eingeladene Benutzer geschickt, so dass die Einladung als Autorisierung, den Chat zu betreten, gesehen werden kann.
Dabei ist es gewollt, dass jeder den Schlüssel weitergeben und somit weitere Benutzer einladen kann.

Dateiübertragungen

Dateien und Bilder kleiner als 32 KB werden wie bei den vorherigen DG Network-Versionen direkt an die Mitteilung angehängt. Beim Umsetzen der Mitteilung in ein Packet werden die Daten dann mit in das Paket eingefügt, so dass das Bild zusammen mit dem gesendeten Text übertragen wird.
Bei Dateien größer als 32 KB würde durch das Anhängen an ein Paket das Netzwerk überlastet werden (ein Paket ist immer ein Block bei der Übertragung, bei einer 100 KB-Datei über eine 8 KB/s-Verbindung müssten alle anderen also 12,5 Sekunden warten), also werden diese über eine neue TCP-Verbindung getrennt verschickt.
Dazu werden die Dateien beim Versenden über den in DG Network eingebauten Webserver unter einem speziellen Code freigeben. Dieser Code wird dann den Empfängern des Chatpaketes mitgeteilt und diese können sich die Datei abholen. Bei Bildern passiert dies automatisch bei der Anzeige im Chatfenster, bei Dateien erst, wenn der Empfänger die Übertragung durch Anklicken der Datei bestätigt.
Da nicht jeder wegen Routern etc. eingehende Verbindungen aus dem Internet annehmen kann, teilen sich beide Seiten über das P2P-Netzwerk gegenseitig ihre IP-Adresse und Port mit und versuchen dann abwechselnd, die direkte Verbindung aufzubauen. Auf dieser Verbindung wird die Datei dann mit den http-Funktionen im Webserver-Modul übertragen. Dabei wird über den Range-Header in http auch Anhalten und Fortsetzen der Übertragung unterstützt. Wenn also eine große Datei übertragen werden soll, kann die Übertragung angehalten und Tage später noch fortgesetzt werden (auch wenn sich bis dahin die IP-Adressen geändert haben).
Bilder werden so wie Dateien verschickt, nur dass sie beim Empfang nicht als Downloadlink sondern direkt als Bild dargestellt werden.

Abbildung 9: Dateien und Bilder verschicken

Weitere Features

DG Network enthält noch einige weitere kleine Funktionen, ich führe hier nur einige davon auf.
Im Chatfeld kann der Benutzer zum Beispiel mit den Tasten hoch/runter zuvor gesendeten Mitteilungen wieder abrufen und so verändern und erneut senden.
Dateien und Bilder können zum Versand nicht nur über die Menübefehle oder Drag'n'Drop aus dem Windows Explorer eingefügt werden, sondern auch über die Zwischenablage. es können nicht nur Bilddateien, sondern auch Bilddaten (z.B. durch Markieren eines Ausschnitts in Paint und Drücken von Strg-C oder durch die „Druck"-Taste für Screenshots) eingefügt werden (diese werden automatisch als .png oder .jpg gespeichert). Beim Einfügen von Bilddaten zeigt DG Network einen Dialog, in dem der zu sendende Ausschnitt gewählt werden kann. Dies ist besonders praktisch beim Senden von Screenshots.

Abbildung 10: Bildausschnitte senden

Smilies können über bestimmte Codes eingefügt werden, bei der Anzeige werden diese dann in Bilder umgewandelt.
Es können mathematische Formeln eingefügt werden, diese werden dann über die MathML-Unterstützung des verwendeten Browsers (der Internet Explorer benötigt dazu ein Plugin, Mozilla kann standardmäßig MathML) angezeigt.

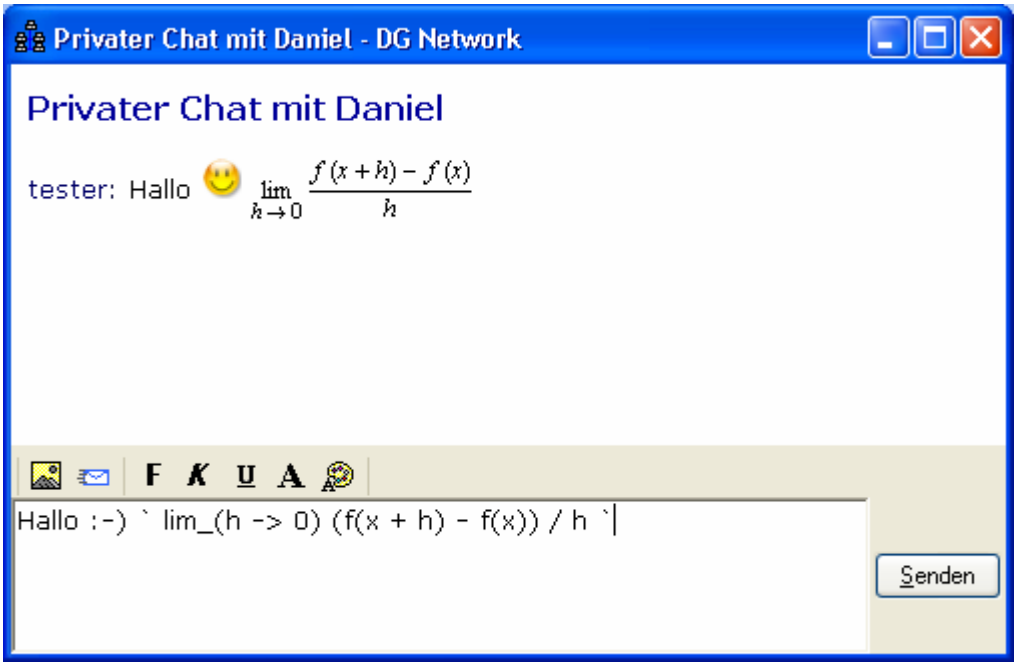

Abbildung 11: Smilies und mathematische Formeln

Die Umwandlung von Smilies und Formeln geschieht in einem speziellen Schritt im HTML-Parser (siehe Seite 11), dabei werden zusammenhängende Textblöcke (also ohne HTML-Tags dazwischen) auf Smilies-Codes oder in Backticks (`) eingeschlossene Formeln überprüft.
Die Formel werden von DG Network in MathML konvertiert und von dem Browser angezeigt. Die Formeln können als einfacher Text eingegeben werden, hier einige Beispiele:

`x^2+y_1+z_12^34`	$x^2 + y_1 + z_{12}^{34}$
`(2a + b*c)/d`	$\frac{2a + b \cdot c}{d}$
`f(x)=sum_(n=0)^oo(f^((n))(a))/(n!)(x-a)^n`	$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$
`sqrt sqrt root3 x`	$\sqrt{\sqrt{\sqrt[3]{x}}}$
`(a,b]={x in RR : a < x <= b}`	$(a, b] = \{x \in \mathbb{R} : a < x \leq b\}$

Abbildungsverzeichnis

Verwendeter Code

Bibliotheken / Tools:
log4net - Logging Framework for .NET - http://logging.apache.org/log4net/
NUnit - Unit testing for .NET - http://www.nunit.org/
NAnt - A .NET Build Tool - http://nant.sourceforge.net/
CommandBar - von Lutz Roeder - http://www.aisto.com/roeder/dotnet/

In DG Network integrierter Programmcode:
SharpDevelop Core - http://www.sharpdevelop.com/
MycroXaml – "A Declarative Xml Parser In Less Than 300 Lines Of Code" von Marc Clifton
http://www.codeproject.com/dotnet/MycroXaml.asp
ASCIIMathML: Translating ASCII math notation to Presentation MathML
http://asciimathml.sourceforge.net/
IconToBitmap with alpha channel
http://dotnetrix.co.uk/misc.html
"Never-ending Progress Bar (C#)" von Greg Osborne
http://www.codeproject.com/cs/miscctrl/C_NeverEndingPBar.asp
"TreeView with Paint event" von J. Young
http://www.codeproject.com/cs/miscctrl/genmissingpaintevent.asp
"Updated Extended ListView" von Bill Seddon
http://www.codeproject.com/cs/miscctrl/Extended_List_View_2.asp
"SharpZipLib"
http://www.icsharpcode.net/OpenSource/SharpZipLib/
"SmartIrc4net - C# IRC library"
http://smartirc4net.meebey.net/

Mein Projekt im Internet: www.danielgrunwald.de/network

Daniel Grunwald

Daniel Grunwald
Bierberger Str. 9
31174 Schellerten

daniel@danielgrunwald.de
www.danielgrunwald.de