DG Network

Programmierung eines P2P-Chats

Inhaltsverzeichnis

Die „Vorgeschichte“

Die Idee

Ich habe in vergangener Zeit schon einen Chat programmiert, mit dem man sich im LAN unterhalten und Nachrichten schicken konnte.
Um im Internet mit meinen Freunden zu chatten, habe ich bisher ICQ verwendet. ICQ hat zwar erweiterte Funktionen um sich z.B. Dateien zuzuschicken oder mit mehr als 2 Usern zusammen zu chatten, jedoch versagen diese Funktionen sobald einer der Teilnehmer einen Router zur Einwahl ins Internet verwendet.
Außerdem störten uns die von Version zu Version immer größer werdenden Werbebanner.
Daher habe ich beschlossen, meinen Chat so umzubauen, dass er für das Internet tauglich ist.
Nach kurzer Zeit stellte ich fest, dass dies nicht so einfach möglich war, denn mein Programm sollte auch bei der Verwendung von Routern noch funktionieren. Dazu kam, dass ich mir die neue Entwicklungsumgebung „Visual Studio .NET“ von Microsoft gekauft hatte.
Daher beschloss ich, eine komplett neue Anwendung mit Visual Basic.NET zu schreiben.
Dabei lernte ich viel über .NET, jedoch funktionierte die Anwendung nicht korrekt, häufig brachen die Verbindungen grundlos zusammen.

Neubeginn

Daher hab ich noch einmal alles neu programmiert, diesmal mit dem Wissen über .NET.
Das hierbei entstehende Programm funktionierte sehr gut.
Alle User, die online waren, verbanden sich mit dem festen „IP-Server“. Dies war eine einfache PHP-Seite auf meinem Webspace. Die Seite wählte aus allen Usern einen aus und teilte dies allen anderen mit. So verbunden sich alle User mit dem „Haupt-User“. Dieser leitete Pakete weiter und verteilte normale Chat-Pakete an alle. Ich habe viele weitere Funktionen wie Dateiübertragungen usw. hinzugefügt.

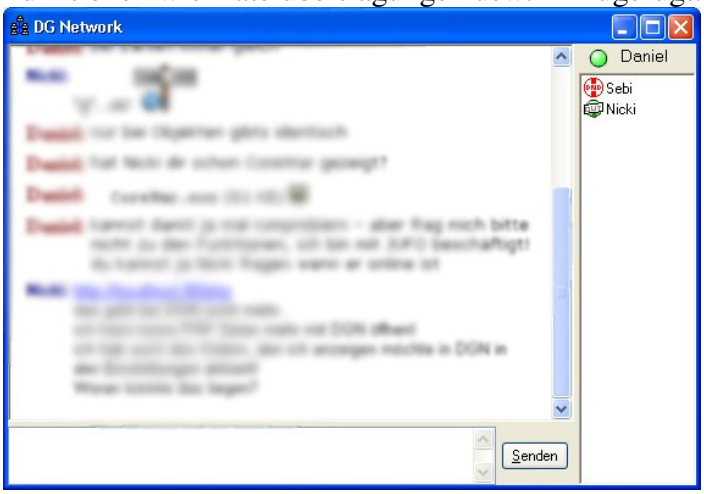
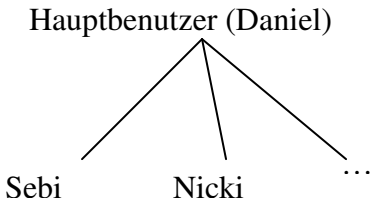
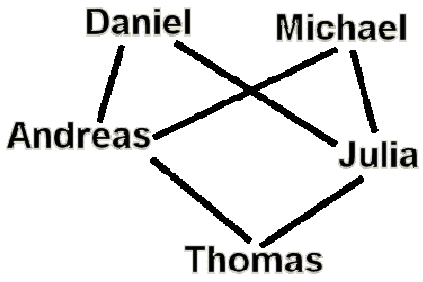


Abbildung 1: die alte Version



Jedoch hatte diese Version nur einen einzigen Chatraum, und da alle User sich mit dem „Haupt-User“ verbunden haben konnten nicht viele das Programm nutzen, ohne dass der Haupt-User überlastet wurde.
Daher fing ich am 29. November noch mal komplett von vorn an. Da Visual Basic.NET einige Funktionen für Multithreading nur umständlich unterstützt, habe ich die neue Version in C# geschrieben.

Planungen für das P2P-Netz



P2P heißt Peer to Peer. Das heißt, jeder User verbindet sich mit ein paar anderen, und insgesamt entsteht ein Netz. Wenn man ein Paket an jemanden schickt, mit dem man nicht direkt verbunden ist, muss man den schnellsten Weg suchen um darüber das Paket zu schicken. Wenn Thomas im Beispiel an Michael schreibt, muss das Paket über Andreas oder Julia weitergeleitet werden.

Warum P2P?

Nun, wenn man von dem höheren Programmieraufwand bei einem P2P-Netz absieht, kann man die Frage auch andersrum stellen:
Warum einen Server zwischenschalten, wenn sich die Benutzer doch direkt unterhalten wollen?
Ein Server kann ausfallen und viele Benutzer verursachen viel Traffic – und das kostet.
Wenn die Benutzer sich untereinander verbinden fällt zwar häufiger mal ein Teil aus wenn jemand offline geht, aber andere Verbindungen können weiter genutzt werden.
Gegen einen festen Server sprechen hauptsächlich die Kosten, denn ein Chat, der Bilder verschicken kann, verursacht einiges an Traffic. Da ist es sinnvoll, diesen Traffic auf viele Benutzer aufzuteilen.
Es gibt aber auch Risiken, zum Beispiel, wenn sich die Verbindungen so anordnen, dass zwei Gruppen getrennt sind. Wenn wichtige Knoten die Verbindung zum Internet trennen, kann es passieren, dass sich das Netz in kleine Teile aufteilt. Es ist also wichtig, dass das Netzwerk entdeckt, wenn es aufgeteilt wurde, und die Teile wieder miteinander verbindet. Außerdem muss sich das Netz so aufbauen, dass die Wege nicht zu lang werden, denn jeder Punkt, wo das Paket umgeleitet wird, erhöht das Risiko, dass es verloren geht, außerdem sind Weiterleitungen im Prinzip unnützer Datenverkehr. Deshalb muss das Netz beim Chatten mit „weit entfernten“ Partnern Verbindungen aufbauen, so dass alle benutzten Verbindungen möglichst kurz bleiben.

IP-Server

Aber wenn ein Programm noch niemanden kennt, weil es grade erst gestartet wurde, muss es zunächst die anderen Benutzer finden. Dazu braucht es einen festen Einwahlknoten. Das ist der oben genannte „IP-Server“. Er macht nichts weiter als bei einer Anfrage die IP-Adresse des Users abzuspeichern und die Adressen der anderen User zurückzugeben.
Damit das Netz auch funktioniert wenn ein IP-Server nicht erreichbar ist, verwendet DG Network zwei Server. Der erste liegt auf meiner Homepage (www.danielgrunwald.de), der andere auf der Homepage meines Freundes Matthias (www.htwc.de).

Sicherheit

Für DG Network ist ein Sicherheitssystem geplant, allerdings habe ich dies bisher noch nicht vollständig umgesetzt.
Zu dem Sicherheitssystem gehört zum einen, dass jeder Benutzer sich beim IP-Server anmelden muss. Er bekommt einen Benutzeraccount und muss diesen per E-Mail bestätigen. Nur über diesen Account kann er den IP-Server ansprechen. Bei der Verschlüsslung des Passwortes verwende ich den Secure Hash Algorithm. Dieser berechnet aus einem Text eine Bytefolge, die sich nicht in den Originalwert zurückwandeln lässt. Die Bytefolge wird normalerweise hexadezimal dargestellt. Es ist nicht nur unmöglich den Originalwert zu ermitteln, es ist auch nicht möglich, einen anderen Text zu finden der den gleichen Hash hat.

Die einzige Möglichkeit wäre Probieren, aber selbst moderne Rechner brauchen Jahre um ein mehrfach verschlüsseltes Passwort durch Probieren zu knacken. So wird immer nur eine verschlüsselte Variante des Passwortes übertragen, der IP-Server verschlüsselt das auf seiner Seite gespeicherte Passwort und vergleicht die verschlüsselten Varianten.
Als ein „Sicherheitsloch“ in DG Network habe ich es angesehen, dass jemand unter falscher UserID Verbindungen aufbaut. Hier lässt sich der Secure Hash Algorithm gut anwenden: Beim Aufbau der Verbindung wird zu dem eigentlichen Passwort die IP-Adresse des Verbindungspartners (per Stringverkettung) hinzugefügt, das ganze wird dann mit SHA-1 verschlüsselt. So kann der Verbindungspartner nicht das ursprüngliche Passwort herausfinden. Der Verbindungspartner kann jetzt vom IP-Server dieses Passwort prüfen lassen. Allerdings kann er den Code nicht verwenden, um sich bei anderen einloggen, da er hierfür das Passwort mit einer anderen IP-Adresse kombinieren müsste. So wird garantiert, dass sich niemand unter falschem Namen einloggt.
Trotzdem bleiben einige Probleme in der Sicherheit des P2P-Netzes, daher habe ich geplant, auch für die Chat-Pakete eine Verschlüsselung zu benutzen. Die Schlüssel dafür werden mit einem asynchronen Verfahren wie RSA übertragen oder bei Chat-Gruppen einfach vom Gruppen-Server zurückgegeben.

Das Netzwerk

Verbinden

Zunächst habe ich im neuen DG Network die Grundstruktur des Netzwerks programmiert. Um den Zustand des Netzes anzuzeigen, habe ich den „Monitor“ programmiert. Angenommen, zunächst ist nur Martin online. Dann starte ich mein Programm. Zunächst muss das Programm die IP-Adresse eines anderen Benutzers herausfinden. Dazu verbindet es sich mit dem IP Server. Dieser liefert die IP-Adresse von Martin zurück. DG Network speichert alle IP-Adressen für 10 Minuten im sog. „Host Cache“ ab. Solange man weniger als zwei Verbindungen hat, versucht DG Network neue Verbindungen aufzubauen. Dazu werden die gespeicherten Adressen im Host Cache verwendet. So baut das Programm nun eine Verbindung zu Martin auf. Danach geht Andreas online. Er fragt auch den IP-Server ab und bekommt die Adressen von mir und Martin und verbindet sich mit beiden.

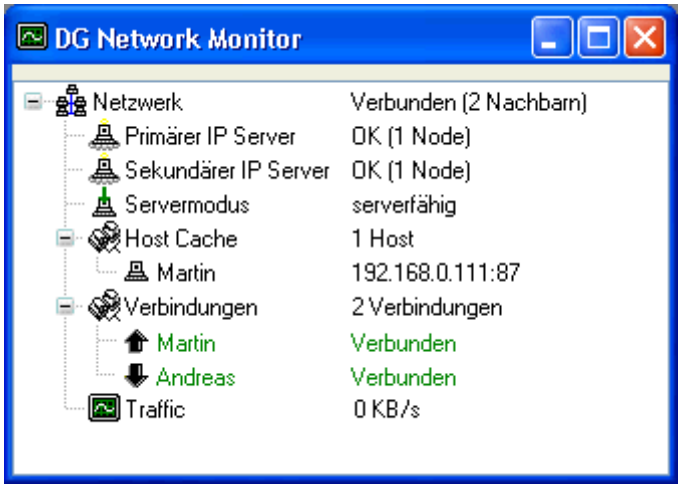


Abbildung 2: DG Network Monitor

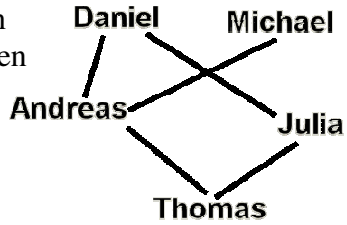
Verbindungen annehmen

DG Network muss natürlich auch in der Lage sein, die Verbindungsanfragen der anderen Benutzer anzunehmen. Dazu wartet DG Network auf einem einstellbaren Port auf eingehende Verbindungen. Welcher Port verwendet wird, teilt DG Network dem IP Server mit, so dass andere Benutzer diesen Port finden können. Hier bereiten die Router ein Problem: wer einen Router hat und hier kein Port Forwarding einstellen will/kann, der kann keine eingehenden Verbindungen annehmen, sondern nur Verbindungen nach außen aufbauen. Das reicht auch aus. Beim Programmstart nimmt DG Network an, dass ein Router verwendet wird, DG Network also nicht serverfähig ist. Beim ersten Abruf des IP-Servers teilt er diesem mit, dass er nicht serverfähig ist. Der IP-Server übermittelt zusätzlich zur Liste mit den IP-Adressen der anderen die IP-Adresse des Benutzers selbst. Wenn ein Router verwendet wird, kennt der Computer gar nicht die IP-Adresse, unter der er im Internet erreichbar ist, sondern nur die im

LAN. Dadurch dass der IP-Server die eigene IP zurückgibt, kennt DG Network nun die eigene externe IP. Nun versucht DG Network, sich mit sich selbst zu verbinden. Wenn dies gelingt, wird kein Router verwendet oder der Router ist passend eingestellt. DG Network wechselt in den serverfähigen Modus. Beim nächsten Abruf des IP-Servers teilt DG Network dem Server mit, das es sich nun als serverfähig einstuft. Dadurch gibt der IP-Server die Adresse des Benutzers an andere weiter.

Das Paketsystem

Ein zentraler Teil in DG Network ist das Paketsystem. Das Paketsystem sorgt dafür, dass ein Paket den besten Weg zum Ziel findet und weitergeleitet wird.
Nun, nehmen wir mal ein Beispiel im Netzaufbau. Wenn Thomas nun Michael ein Paket schickt, gibt es verschiedene Möglichkeiten zum Ziel zu kommen. Wir suchen den kürzesten Weg, allerdings kennt jeder nur seine Nachbarn, nicht den gesamten Aufbau des Netzes. So können wir keinen Weg finden. Jeder müsste seinen Nachbarn mitteilen, wen er kennt, dieser müsste das dann wieder weitersagen usw.



Routing-Tabellen

Jeder soll seinem Nachbarn sagen, wen er erreichen kann.
D.h., wenn die Verbindung Thomas-Andreas erst zum Schluss aufgebaut worden ist, wären dort folgende Tabellen gesendet worden:
Thomas sendet an Andreas: Ich kenne Julia (Abstand 1), Daniel (Abstand 2) und Michael (Abstand 4)
Andreas sendet an Thomas: Ich kenne Daniel (Abstand 1), Michael (Abstand 1) und Julia (Abstand 2)
Thomas wertet diese Tabelle aus:
Andreas kennt Daniel im Abstand 1, über Andreas würde der Weg also 2 lang sein. Der bisherige Weg über Julia ist auch 2 lang, also wird dies nun als Alternativ-Route eingetragen.
Andreas kennt Michael im Abstand 1, also wäre der Weg über Andreas 2 lang. Das ist kürzer als der bisherige Weg, also benutzt Thomas in Zukunft den Weg über Andreas. Schließlich kennt Andreas noch Julia im Abstand 2, also wäre der Weg 3 lang. Der bisherige Weg ist nur 1 lang, also wird der bisherige Eintrag beibehalten.

Andreas wertet nun die Tabelle, die er von Thomas erhalten hat, aus.
Thomas kennt Julia mit Abstand 1, also wäre der Weg 2 lang. Dies ist eine Alternative zum Weg über Daniel. Thomas kennt Daniel mit dem Abstand 2, über Thomas würde der Weg also 3 lang sein. Da ist der vorherige Eintrag mit dem Abstand 1 kürzer. Und Thomas kennt Michael mit dem Weg 4, da wäre der Weg 5 lang. Da ist die Direktverbindung klar kürzer, also wird der alte Eintrag behalten.

Alle Änderungen durch Hinzufügen der Verbindung Andreas-Thomas auf einen Blick:
Thomas -> Daniel: Alt über Julia (2); Alternative über Andreas (2)
Thomas -> Michael: Alt über Julia (4); Neu über Andreas (2)
Andreas -> Julia: Alt über Daniel(2); Alternative über Thomas (2)
Ohne dass dies in den Tabellen übertragen wird fügt jeder natürlich noch den Direktweg in seine Tabelle ein:
Thomas -> Andreas: Alt über Julia (3); Neu direkt (1)
Andreas -> Thomas: Alt über Daniel (3); Neu direkt (1)

Beide prüfen nun, ob sich durch diese Änderungen Vorteile für Ihre Nachbarn ergeben haben.

Michael -> Thomas war vorher 4 lang, jetzt nur noch 2. Andreas sendet die verbesserte Tabelle an Michael, damit Michael den Längeneintrag korrigieren kann. So wird eine Verbesserung immer weitergegeben. Kommt ein neuer User ins Netz, so wird er auch mit dieser Methode weitergegeben. Eine andere Funktion sorgt dafür, dass beim Zusammenbruch der Verbindung alle Wege, die diese Verbindung nutzen, entfernt werden.

Die Pakete

So hat nun jeder eine Tabelle, in der steht, zu welchem User welche Verbindung optimal ist. Damit können die Pakete zu ihrem Ziel weitergeleitet werden.
Aber was ist nun ein Paket genau?
Ein Paket umfasst folgende Informationen:

- Angabe zum Pakettyp. Dadurch werden verschlüsselte und versicherte Pakete möglich
- Länge und eine Prüfsumme des Datenteils des Paketes
- Quelladresse.
- Zieladresse(n)
- Datenteil

Eine Adresse besteht aus zwei Teilen. Der erste Teil ist die UserID, eine eindeutige Nummer mit der jeder User identifiziert werden kann. Der zweite Teil ist eine Programmnummer. Sie gibt an von welchem Programmteil das Paket beim Ziel verarbeitet werden soll. So kann zwischen Paketen für den Chat und Paketen für die Benutzerliste unterschieden werden.
Ein Paket kann auch an mehrere Ziele geschickt werden. Dann wird das Paket bei jeder Zwischenstation kopiert, wenn nötig. Thomas sendet z.B. ein Paket an Julia, Daniel und Michael. Als Weg zu Daniel wird jetzt der Weg über Andreas ausgewählt, es könnte auch andersherum sein, das macht keinen Unterschied. Thomas sendet das Paket nun an Andreas mit dem Vermerk „für Michael und Daniel“ und an Julia mit dem Vermerk „für dich“.

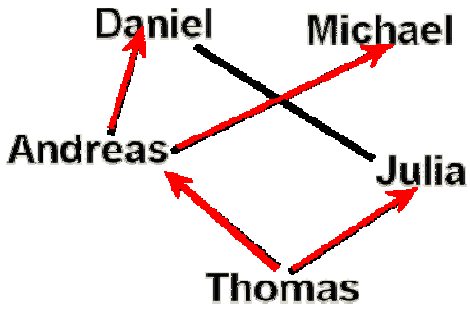


Abbildung 3: Weg eines Paketes

Das Weiterleitungssystem

Wie schafft es das Programm jetzt, in den Tabellen den besten Weg zu finden und das Paket entsprechend zu kopieren?
Dazu hab ich mir beim Programmieren überlegt: „Die Post führt einen neuen Dienst ein: man kann Postkarten an mehrere Leute gleichzeitig schicken“.
Jede Poststelle hat Verbindungen zu einigen anderen Poststellen und zu den Benutzern selbst. Die Poststelle will Papier und Weg sparen, also wird nur so wenig wie möglich kopiert. Die anderen Poststellen können dann ja weiter kopieren. Am wichtigsten ist, dass der gesamte Transportweg möglichst kurz ist.
Zunächst nimmt der Postbeamte also den Adresszettel. Jetzt geht er jede einzelne Zieladresse durch. In der Routing-Tabelle sucht er heraus, wie er diese Adresse erreichen kann. Das schreibt er als Notiz hinter den Adresszettel. Wenn eine Adresse auf ihn selbst zeigt, kopiert er sich die Karte und wertet die Daten aus. Wenn er eine Zieladresse nicht finden kann, streicht er die Adresse durch, war es ein gesichertes Paket wird es als unzustellbar an den Absender zurückgeschickt.
So hat er jetzt hinter jeder Adresse stehen, an welche Poststelle er die Karte weiterleiten muss. Er geht alle Poststellen, mit denen er verbunden ist durch und sucht die entsprechend markierten Adressen heraus. Für jede Poststelle, die mindestens einmal auf der Adressliste notiert ist, kopiert er die Postkarte einmal und sendet sie dorthin, und sendet ihr alle Adressen, die entsprechend markiert waren, als neue Ziele.

So wird das Paket weitergeleitet und immer weiter aufgeteilt, bis es am Ziel ankommt oder als unzustellbar zurückgesendet wird.

Benutzergruppen

Eine Gruppe ist in DG Network einfach eine Liste von Benutzern. Andere Benutzer können sich in diese Liste einfügen, austragen und die Liste anzeigen.

Freie Gruppen

Als erstes habe ich in DG Network die „freien Gruppen“ programmiert. Dies sind Gruppen ohne feste Mitglieder. Jeder kann sich in eine solche Gruppe einwählen. User, die nicht online sind, werden automatisch aus der Gruppe entfernt. In seiner Gruppenliste kann man freie Gruppen einfach eintragen. Eine Gruppe wird dabei durch die eindeutige Gruppennummer festgelegt. Wenn man jetzt mit dieser Gruppe chatten möchte, muss DG Network herausfinden, wer sich im Moment in dieser Gruppe befindet. Dazu gab es zwei Möglichkeiten, zwischen denen ich wählen musste:
Ich könnte in das Netzwerk eine Art „Suche“ einbauen; so wie andere Netze für Quellen bei Dateidownloads suchen, könnte ich nach Usern in einer Gruppe suchen. Allerdings passiert es relativ häufig, dass man eine Gruppe betritt/verlässt, und das gesamte Netz zu durchsuchen, ist zu aufwendig. Bei P2P-Filesharing-Programmen ist es so, dass nur eine unvollständige Suche durchgeführt wird, denn einige Quellen reichen für eine Dateiübertragung aus. Für mein Chat-Netzwerk benötige ich hingegen eine vollständige Suche. Daher habe ich mich für die zweite Möglichkeit entschieden: So wie der IP-Server die IP-Adressen verwaltet, könnte ein Gruppen-Server die Gruppen verwalten. Dies wäre auch nützlich bei den anderen Gruppentypen, wo nur bestimmte User eine Gruppe betreten dürfen. Daher sollte der Gruppen-Server zusammengelegt werden mit dem IP-Server, wo schon die Benutzerdaten verwaltet sind. Gruppen-Server und IP-Server sind zwei PHP-Seiten auf meiner Homepage.

Abbildung 4: Gruppensuche

Also habe ich einen kleinen Gruppen-Server programmiert. Außer der Abruf-Funktion gibt es auch die Möglichkeit neue Gruppen anzulegen und vorhandenen Gruppen beizutreten. Dies findet vollständig im Browser statt. Wenn man einer Gruppe beitritt, wird diese einfach in die Gruppenliste hinzugefügt. Dadurch ist man noch kein Mitglied in der Gruppe, das ist bei freien Gruppen auch gar nicht nötig.

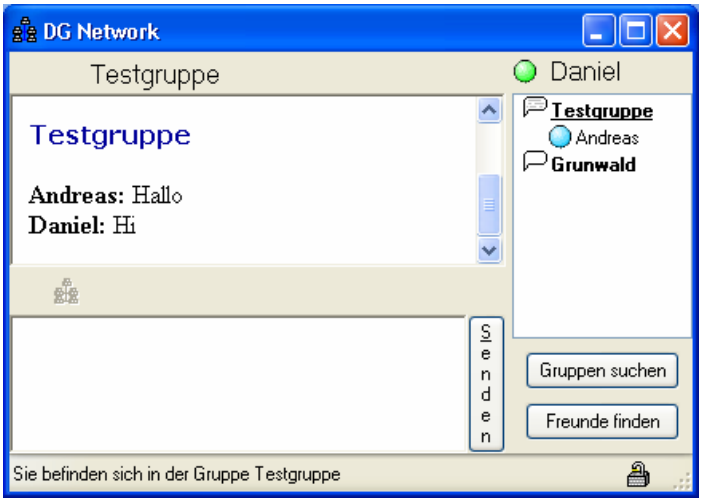


Abbildung 5: Gruppenauswahl

Wenn man nun per Doppelklick einer freien Gruppe beitritt, ruft DG Network vom Gruppen-Server die Benutzerliste ab. An diese Benutzer wird nun eine Anfrage gesendet. Die Benutzer, die immer noch in dieser Gruppe sind, antworten auf die Anfrage und fügen uns zu der Gruppe hinzu. Wenn der andere Benutzer noch weitere Gruppenmitglieder kennt schickt DG Network auch diesen noch eine Anfrage.

Ein Benutzer in einer freien Gruppe kann nur vier Zustände haben:
- Der Benutzer hat momentan das Chatfenster mit dieser Gruppe ausgewählt.
- Der Benutzer ist in dieser Gruppe, hat aber ein anderes Fenster ausgewählt.
- Der Benutzer ist für kurze Zeit nicht am Computer gewesen.
- Der Benutzer ist für längere Zeit nicht am Computer gewesen.

Wenn jemand die Gruppe verlässt/das Fenster schließt, wird er in einer offenen Gruppe nicht mehr als Mitglied gewertet.

Geschlossene Gruppen

Bei einer geschlossenen Gruppe muss man zunächst Mitglied werden, bevor man die Mitgliederliste abfragen darf. Zusätzlich verschlüsselt eine geschlossene Gruppe alle Chat-Mitteilungen. Der Schlüssel wird vom Gruppenserver abgefragt und nur an eingetragene Mitglieder weitergegeben.
Um Mitglied einer geschlossenen Gruppe zu werden, muss man vom Moderator dieser Gruppe zunächst die Erlaubnis erhalten. Der Moderator ist derjenige, der die Gruppe eingerichtet hat, er kann sein Amt aber auch an jemanden anders abgeben.
Mit einer Mitglieds-Erlaubnis kann man seine Mitgliedschaft eintragen und hat daraufhin das Recht, die Mitgliederliste und den Schlüssel abzurufen.
Bei einer geschlossenen Gruppe gibt der Gruppen-Server alle Mitglieder zurück. Man gilt auch als Mitglied, wenn man im Moment nicht in der Gruppe ist. Daher gibt es folgende zusätzliche Zustände:
- Der Benutzer ist Mitglied, ist aber momentan nicht online.
- Der Benutzer ist Mitglied, hat diese Gruppe aber nicht ausgewählt.

In diesen beiden Zuständen empfängt ein Benutzer keine Chatmitteilungen.

Dynamische Gruppen / Privater Chat

Dynamische Gruppen benötigen keinen Gruppen-Server. Man sie auch nicht in die Gruppen-Liste eintragen, da sie keine Einwahl-Möglichkeit bieten.
Dynamische Gruppen werden bei bestimmten Aktionen automatisch erzeugt – zum Beispiel wenn Sie auf einen Benutzer im Kontext-Menü den Befehl „Privater Chat“ wählen. Dabei wird eine dynamische Gruppe mit einer zufälligen Gruppen ID erzeugt. Diese Gruppe existiert zunächst nur auf dem PC des Benutzers der den Chat einleiten will. Wenn der Zielbenutzer der Einladung zustimmt, sendet er dem anderen ein Paket, um dort der Gruppe beizutreten, und er legt auch bei sich das Gruppenobjekt an.
DG Network hat nun die Funktion, das einer der beiden im privaten Chat per Drag'n'Drop einfach einen weiteren Benutzer in den Chat ziehen an. Dadurch wird auch diesem die Nummer des privaten Chats mitgeteilt, wenn er die Einladung akzeptiert verbindet er sich zunächst mit dem Benutzer der ihn eingeladen hat, dieser teilt ihm alle anderen Benutzer die sich in dieser dynamischen Gruppe befinden mit.

Freunde

Die Liste der Freunde ist auch eine Gruppe, da Gruppe als Liste von Benutzern definiert ist. Allerdings kann man mit dieser Gruppe nicht chatten, man kann nur einzelne User aus dieser Gruppe in einen privaten Chat einladen. Die Benutzer in dieser „Gruppe“ werden einfach in der DG Network Konfiguration gespeichert. Allerdings ist diese Gruppe nicht vollkommen ohne Funktion: DG Network fragt auch hier die anderen User an, um ihren Status zu empfangen.
Die bisher gezeigten Lampen haben hier eine leicht geänderte Bedeutung, und es gibt viele weitere Statussymbole.

 Der Benutzer momentan nicht online.
 Der Benutzer hat ein Chatfenster geöffnet und markiert.
 Der Benutzer hat ein Chatfenster geöffnet, es ist aber nicht markiert.
 Der Benutzer hat kein Chatfenster geöffnet.
 Der Benutzer ist für kurze Zeit nicht am Computer gewesen.
 Der Benutzer ist für längere Zeit nicht am Computer gewesen.
 Der Benutzer möchte nicht gestört werden.
Weitere Symbole sind in Planung.

Der Chat

Der Chat ist im Prinzip das Fenster, mit dem man chattet. Zunächst muss man eine Gruppe/einen Freund wählen. Dann kann man mit diesem chatten.
Das Chatfenster ist noch nicht ganz fertig gestellt, in dem grauen Balken zwischen Eingabefeld und Anzeige ist eine Toolbar mit wichtigen Befehlen wie Flüstern, Bild einfügen und Smilie-Wahl vorgesehen.
Das Eingabefeld ist eine RichTextBox, d.h. eine Eingabebox die durch Code im RTF Format den Text formatiert darstellen kann und Bilder einfügen kann. Das Feld wo man die Einträge der anderen sieht, ist ein Webbrowser.

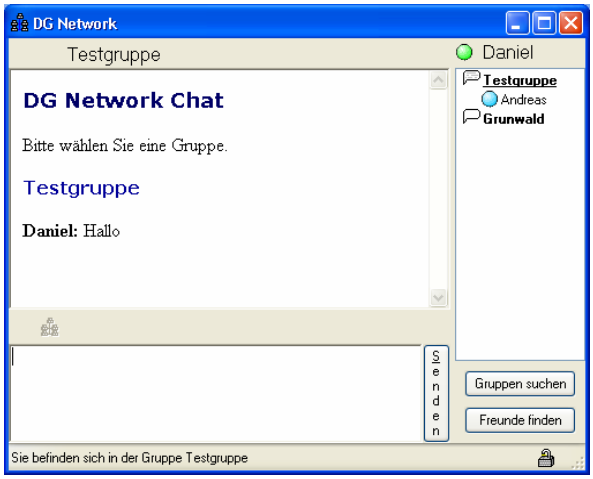


Abbildung 6: Im Chat

Chatbrowser

Dieser Webbrowser ist einfach der Microsoft Internet Explorer über COM in meine Anwendung eingebunden. Dadurch kann ich viele Dinge im Chat darstellen: Formatierten Text, Bilder und Hyperlinks.
Ich habe hier keine RichTextBox verwendet, da ich Funktionen benötigte, die RTF nicht hat. Zum Beispiel kann RTF keine animierten Bilder, die aber bei vielen Smilies nötig sind. Mit HTML habe ich viel bessere Möglichkeiten, z.B. anklickbare Bilder bei bestimmten Aktionen.
Nun habe ich mich also für HTML entschieden. Es ist kein Problem mit dem Internet Explorer eine Seite im Internet abzurufen und in meinem Programm anzuzeigen. Aber wie sollte ich meine eigenen Daten in den Browser bekommen?
Vielleicht gibt es eine Möglichkeit, aber ich hab nie lange gesucht. DG Network konnte Verbindungen akzeptieren, und bei Verbindungen zu localhost, also der eigene PC, funktioniert dies immer – auch bei Routern. So lasse ich den Internet Explorer auf die Seite http://localhost:port/chat/stream surfen. Die Verbindung wird daraufhin vom Servermodul, das auch Netzwerk-Verbindungen annimmt, akzeptiert. Das Modul stellt fest, dass es eine HTTP Anfrage ist, und leitet die Anfrage an den Webserver weiter. Der Webserver ist ein kleines Modul in DG Network, das ich vielleicht irgendwann in einen richtigen Webserver erweitern werde. Vorerst macht er nichts weiter als alle Anfragen auf /chat/stream an das Chatmodul zu leiten und die restlichen Anfragen zu ignorieren. Das Chatmodul prüft jetzt, ob die Quelle des Aufrufs der eigene PC ist. Falls nicht wird die Verbindung getrennt. So wird gesichert, dass sich niemand von außerhalb die Chateinträge anzeigen lassen kann.
Die Tatsache, dass ich die Chateinträge per Webserver verschicke hat mich auf eine andere Idee gebracht, die ich aber erst später programmieren werde: **WebUser**
Ein WebUser ist ein Benutzer, der auf seinem Computer kein DG Network laufen hat, sondern sich mit einem Browser bei einem anderen DG Network Benutzer „eingeklinkt“ hat. Dort bekommt er ein eigenes User-Objekt und eine eigene UserID. Damit er dort auch Pakete

empfangen kann, wird er in der Routing-Tabelle eintragen, als wäre er ein richtiger Benutzer, der sich hier mit dem Netzwerk verbunden hat. Dadurch kann ein WebUser wie jeder andere Pakete senden und empfangen. Jedoch müsste ich die Funktionalität der restlichen Benutzeroberfläche (Gruppenwahl, Eingabefeld usw.) für WebUser anpassen, daher wird es WebUser nicht in nächster Zeit geben.

Der Chatcode-Parser

Ein Benutzer darf in seinem Eingabefeld HTML-Anweisungen benutzen, um seinen Chateintrag zu formatieren. <b>, <i>, <u> und <img>-Tag sollten erlaubt sein. Tags wie <script> oder <object> sollten aber verboten sein, da man unerwünschte Elemente schicken könnte oder auf diese Weise sogar Sicherheitslöcher nutzen könnte, denn http://localhost liegt im „Lokalen Intranet“ und hat damit höhere Rechte als normale Internet-Webseiten.
Also muss ich alle gefährlichen HTML-Tags filtern und nur die „guten“ zulassen.
Da es hunderte gefährliche HTML-Tags und Attribute (z.B. <img onMouseOver=“…“>) gibt, ist es sinnvoller nur die ungefährlichen Tags zuzulassen.
In der Chatbox ist also kein HTML erlaubt sondern nur „Chatcode“. Mein Chatcode ist eine Mischung aus HTML und dem Code, wie er in vielen Foren verwendet wird. Also müssen sowohl <img src=“http://…“> sowie [img]http://…[/img] und [img=http://…] zugelassen werden. Dazu habe ich einen HTML-Parser geschrieben, der auch die „Spezialitäten“ des Forencodes wie [img=URL] unterstützt.
Der Parser läuft in mehreren verschachtelten Schleifen und wird beim Empfangen einer Nachricht ausgeführt. Es gibt noch einen anderen kleinen Parser, der beim Senden ausgeführt wird und die Smilie-Codes in <img>-Tags umwandelt.

Textebene	**Tagebene**	**Parameterebene**	**Werteebene**
Wenn < oder [gefunden wird, wird Tagebene aktiv.	Parset <* oder [* Taucht nach dem Tagnamen ein = auf, wechselt er direkt in die Werteebene, bei einem Leerzeichen in die Parameterebene. Wertet Tag aus	Liest die Parameternamen aus und ruft bei jedem Wert die Werteebene auf.	Liest den Wert hinter dem = aus, dabei werden " berücksichtigt.
Kopiert alle Zeichen, geschützte Zeichen werden umgewandelt			

Umwandlung von geschützten Zeichen
Hier werden die Zeichen, die von HTML reserviert sind, umgewandelt:
& wird zu & < wird zu < > wird zu > " wird zu "
Zeilenvorschub wird zu
Bei ungültigen Zeichen in einer Ebene bricht die Ausführung der Schleife ab. Dadurch fällt die Ausführung in die Textebene zurück, der gesamte Tag wird als Text geparst. Dadurch wird a < b > c zu a < b > c umgewandelt, denn die Leerzeichen im Tag machen den Tag ungültig. Genauso wird bei ungültigen Parametern der gesamte Tag als ungültig erklärt.
Tag auswerten
Der Tagname und die Parameter wurden ausgelesen, jetzt wird der Tag ausgewertet:
Ist der Tagname unbekannt, wird wie bei ungültigen Zeichen abgebrochen.

DG Network ruft die Funktion „startTag“ auf. Dieser Funktion werden der Tagname und eine Liste der Parameter des Tags übergeben. Nehmen wir mal das Beispiel <img>.
Hier sind folgende Tags möglich: <img src=“URL“>, <img=“URL“>Alt-Text</img> und <img>URL</img>. Dem Tagparser ist bisher aber nur der Starttag bekannt. Daher untersucht der Parser die Parameter des <img>-Tags. Wird der Parameter src verwendet, wird direkt der Zieltag geschrieben. Ist src nicht angegeben, muss zunächst der Inhalt des Tags gelesen werden, egal ob dort Alt-Text oder URL stehen. Daher gibt die Funktion „startTag“ einen Wert zurück: Entweder TagType.Invalid, TagType.Simple, TagType.Content oder TagType.Block. Invalid ist ein ungültiger Tag, dies wird z.B. zurückgegeben wenn der Tagname ungültig ist. Dadurch wird der gesamte Tag noch einmal auf der Text-Ebene geparset und so auf der Seite angezeigt. Simple ist ein einfacher Tag wie <img src=“URL“>, es gibt kein Endtag. Content ist ein Tag wie <img>URL</img>: er enthält Text, der Inhalt kann allerdings keine Untertags enthalten. Wird Content zurückgegeben, sucht DG Network den Endtag und ruft die Funktion „endContentTag“ mit dem Tagnamen, der Parameterliste (des Starttags) und dem Inhalt des Tags auf. Anschließend fährt die Textebene mit dem Text hinter dem Endtag fort. Der letzte Tagtyp ist Block. Dies gibt einen Tag an, der Inhalt haben kann und wo der Inhalt weitere Tags enthalten kann. <b> ist ein einfaches Beispiel dafür. Hier verwaltet DG Network alle geöffneten Tags in einem Stack und sorgt dafür, dass auch wenn das Endtag nicht im Code angegeben ist, dafür, dass die Funktion „endBlockTag“ aufgerufen wird.
So wird jeder Tag einzeln analysiert und in HTML umgeformt. Der Benutzer kann so Forencodes wie [img=www.host.de/meinbild.gif]Das bin ich[/img] verwenden und DG Network formt diesen Tag in <img src="http://www.host.de/meinbild.gif" alt="Das bin ich"> um. Wenn noch kein http:// vor einer URL steht wird dies hinzugefügt. Dadurch wird verhindert, dass eine URL per <a href="javascript:"> JavaScript-Befehle auf der Chatseite ausführen kann.
Beim Umformen werden auch nur die bekannten Parameter übernommen – Parameter wie onMouseover, die JavaScript ausführen könnten, werden nicht übernommen.

Bilder im Chat

DG Network bietet mehrere Möglichkeiten, Bilder oder andere Elemente zu verschicken. Zum einen gibt es eine Schaltfläche auf der Toolbar, um Dateien in die Mitteilung einzufügen. Dateien können aber auch per Drag'n'Drop vom Explorer aus in das Chatfeld gezogen werden. Auch können Dateien im Explorer mit Strg-C kopiert und in DG Network mit Strg-V eingefügt werden. All das benutzt die gleiche Funktion: Datei versenden. Aber man kann in einem Bildbearbeitungsprogramm auch einen Ausschnitt eines Bildes markieren und kopieren und diesen dann in DG Network einfügen. DG Network speichert das Bild temporär als JPG ab und verschickt diese Datei.
Beim Verschicken von Dateien gibt es zwei Methoden: entweder die Datei wird direkt im Chat-Paket an alle anderen gesendet oder die Datei wird über den DG Network Webserver den anderen zum Abholen bereitgestellt. Die zweite Methode hat den Nachteil, dass Benutzer mit Router keine Bilder verschicken könnten, die erste Methode hat den Nachteil, dass der gesamte Traffic über das Netz geht. Deshalb habe ich mich für eine Mischung der beiden Methoden entschieden. Bei Dateien kleiner als 32 KB wird die Datei über das Netzwerk verschickt. Bei Dateien größer als 32 KB wird die Datei auf dem „Webserver“ abgelegt und über das Netzwerk wird nur ein <img src="http://IP:port/chat/files/…">-Tag verschickt. In beiden Fällen wird das Bild direkt im Chat der anderen Benutzer angezeigt.

Dateien verschicken

Ich habe geplant, auch andere Dateien als Bilder verschicken zu können. Auch hier werden Dateien kleiner als 32 KB direkt versandt. Bei Bildern größer als 32 KB wurde ein Tag mit

der IP-Adresse verschickt, bei anderen Dateien wird hingegen ein Tag in der Form <file url="dgn:benutzerid/chat/files/…"> verschickt. Jetzt kann der Benutzer erst mal entscheiden, ob er die Datei überhaupt herunterladen will. Wenn ja, fragt DG Network an, welche IP Adresse die Quelle momentan hat und verbindet sich mit ihr. So wird die Datei übertragen. Sollte hier die Quelle einen Router haben, wird die Verbindung in der anderen Richtung aufgebaut. So kann eine Datei übertragen werden solange nur einer von beiden serverfähig ist. Dieses Dateiübertragungssystem gab es schon in der alten Version, in der neuen Version habe ich es noch nicht programmiert.

Plugins

DG Network unterstützt auch Plugins. Dies sind DLLs, die im „Plugin“-Verzeichnis gespeichert sind. Dank .NET Assemblies war die Programmierung des Plugin-Systems relativ einfach. Es funktioniert folgendermaßen:

DG Network.exe Erbt von Plugin.dll	
Plugin.dll Definiert Basisklassen und Schnittstellen	
Plugin Benutzt in der Plugin.dll definierte Funktionen um auf DG Network zuzugreifen.	Weitere Plugins

DG Network benötigt die Plugin.dll.
In der Plugin.dll sind viele Klassen definiert, die DG Network gemeinsam mit den Plugins benutzt.
Beim Programmstart sucht DG Network alle Plugins im Plugin-Verzeichnis. Diese Plugins werden mit vorgefertigten .NET Funktionen geladen. Dann werden die Plugins nach Klassen durchsucht, die von der Schnittstelle “IPlugin“ (definiert in Plugin.dll) abgeleitet sind. Von diesen Klassen wird jeweils eine Instanz erzeugt, dabei werden die Hauptobjekte von DG Network übergeben. Das Plugin kann darauf mit den in der Plugin.dll definierten Funktionen zugreifen und so die Funktionen von DG Network verändern und erweitern.
Plugins haben die Möglichkeit, eigene Einträge im „DG Network Monitor“ (Seite 4) zu erzeugen, sich in den Webserver zu integrieren, Pakete zu senden und Pakete über eigene Programmnummern zu empfangen. Außerdem können Sie vorhandene Elemente auf der Benutzeroberfläche manipulieren und so an vielen Stellen weitere Funktionalität hinzufügen.

Weitere Funktionen

DG Network hat noch viele weitere Funktionen. Die meisten kann man schnell erreichen, indem man ein Kommando in den Chat eingibt. „/ring Benutzername“ lässt z.B. bei dem Zielbenutzer einen Klingel-Sound ertönen. Solche Befehle werden meist durch Plugins hinzugefügt. Wenn man in der Benutzerliste einen Rechtsklick auf einen Benutzer macht, erscheint eine Liste mit den verfügbaren Befehlen.
Ein Beispiel ist das Klingeln: Durch Klingeln-Befehl im Menu wird ein „Klingel-Paket“ an den Benutzer geschickt. Dort wird das Paket ausgewertet, ein Glockenton wird abgespielt und ein kleines Popup erscheint in der rechten unteren Bildschirmecke. Klickt man auf dieses Popup gelangt man direkt in den Chat, von dem aus geklingelt wurde. Wird über die Freundesliste geklingelt wird ein privater Chat aufgebaut.

Abbildung 7: Klingeln

Abbildungsverzeichnis

An dieser Stelle möchte ich mich recht herzlich bei meinem Freund Nicolaj bedanken, der mein Programm ausgiebig getestet und Tipps zur Gestaltung gegeben hat.

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

Daniel Grunwald

Daniel Grunwald
Bierberger Str. 9
31174 Schellerten

daniel@danielgrunwald.de
www.danielgrunwald.de