DIDADIT - Ein neues Dateiübertragungsprotokoll für Packet Radio

Hans-Peter Zorn, DG4IAD@DB0AIS.#HES.DEU.EU
<hp@1409.org>

28. März 2000

1 Entstehung

Schon vor einigen Jahren hatten Jochen, DG6VJ, Walter, DG9EP und ich über einen Ersatz für das Autobin Protokoll diskutiert. Das erste Ergebnis war "EBIN", welches durch unglückliche Umstände damals in die BBS eingespielt wurde und sich dort eine für Packetradio typisch unfruchtbare Diskussion entzündete. Was schliesslich dazu führte, dass Anfang '97 bei db0prt eine Mailingliste eingerichtet wurde, auf der das Protokoll diskutiert und entwickelt wurde. Aktiv beteiligt an der Entwicklung waren Jochen, DG6VJ, Walter, DG9EP, Ulf, DH1DAE, Marco, DL8NJY, Reinald DH2MBM, Hans-Peter, DG4IAD und einige andere.

2 Ziele

Wir waren mit den bisherigen Lösungen wie Autobin, yapp nicht zufrieden. Insbesondere sollte das neue Verfahren folgende Eigenschaften besitzen:

1. Robustheit
2. Unabhängigkeit vom unterliegenden AX.25
3. Möglichkeit, sicher und einfach Dateiübertragungen durch Mitschreiben fremder Verbindungen mit zu lesen und abzuspeichern.
4. Abgebrochene Transfers müssen fortgesetzt werden können.
5. Fehler müssen schon während der Übertragung erkannt und korrigiert werden.
6. Das Protokoll muss leicht erweiterbar sein.

3 Umsetzung

Um Punkt 1 und 2 zu genügen, benutzt das Protokoll Byte-stuffing. Es handelt sich um das selbe Verfahren wie bei SLIP bzw KISS. Da KISS heute kaum noch verwendet wird, sollte das nicht als Overhead ins Gewicht fallen. Der

Datenstrom wird in Blöcke eingeteilt, jeder der Blöcke ist durch das Stuffing gleichermassen aus dem Strom extrahierbar. Es gibt verschiedene Block-Typen:

1. Blöcke, die zu einer bestimmten Verbindung gehören
2. sonstige.

Erstere beinhalten direkt am Blockanfang zunächst einen MD5-Hash der übertragenden Datei. Auf diese Art und Weise soll das Mitschreiben von DIDADIT-Übertragungen vereinfacht werden. Man könnte sich vorstellen, dass irgendwann auch normale Bulletins aus den Mailboxen statt als ASCII-Text per R ausgelesen werden. Dann könnten Programme wie DigiPoint, die ihre lokale Mailbox durch Mitschreiben vom Monitor füllen, weitgehend auf Heuristiken verzichten und die Qualität der Mitschriebe wäre wesentlich höher. Außerdem dient der MD5-Hash zur Intergritätsprüfung, die einzelnen Blöcke sind derzeit nur durch einen CRC-16 gesichert, was in den allermeisten Fällen ausreichend sein sollte. Sollte sich trotzdem noch ein Übertragungsfehler einschleichen, wird dieser sehr sicher über die MD5-Summe erkannt. Die Idee der MD5-Summe als Hilfe zum Mitschreiben stammte übrigens von Ulf, DH1DAE.

Bei einem Verbindungsabbruch kann DIDADIT nach erneutem Connect problemlos die Übertragung fortsetzen. Die Übertragung ist durch den MD5-Hash eindeutig identifiziert und wird mit nach dem letzten Block fortgesetzt. Der CRC befindet sich am Blockende und kann somit on-the-fly berechnet werden.

Übetragungsfehler werden registriert und am Ende der Übertragung en Block korrigiert. Dadurch ist keine aufwendige und verlangsamende end-to-end Bestätigung notwendig.

Am Anfang der Übertragung einigen sich die Gegenstellen auf die Protokoll-Version und die implementierten Erweiterungen. Auf diese Art und Weise ist DIDADIT abwärtskompatibel. Beispielsweise wurde die komprimierte Übertragung bisher nur von WinSTOP implementiert, Transfers zwischen Paxon und WinSTOP Stationen erfolgen eben unkomprimiert.

Ein weiteres Ziel, das wir uns anfangs gesetzt hatten, war Simplizität. Vor allem wollten wir weitgehend auf Timer verzichten, und den Ablauf möglichst eindeutig zu spezifizieren.

4 Implementierung

Ein Protokoll zu spezifizieren ist immer das eine, die Implementierung das andere. Die gilt auch für DIDADIT, mit jeder Implementierung, die dazu kam, wurde das Protokoll angepasst. Die ersten Programme, die das neue Protokoll unterstützten, waren LinKT von DG6VJ und WinSTOP von DL8NJY. WinSTOP hat seine Freunde vor allem im CB-Funk, hier ist geringer Overhead sehr wichtig und Marco hat hier einige Vorschläge gemacht, unter anderem eine Erweiterung um transparent komprimierte Dateien zu übertragen.

Jonny, DH3MB, hat DIDADIT für die BayCom-Mailbox implementiert. Zunächst nur für den Fileserver, später sollen auch BoxBin-Mails ausgelesen werden können und zuletzt auch jede beliebige Nachricht aus der BBS. Bei Mailboxen, die eine Vielzahl von Benutzern bedienen, ergibt sich ein Problem: Aufgrund der Tatsache, dass bei DIDADIT die Fehlerkorrektor am Anschluss an die Übertragung erfolgt, muss für jeden Upload zur BBS eine Protokolldatei geführt werden, die Informationen über den Transfer enthält. Bei Mailboxen stellt sich die Frage, wie lange solche Protokolldateien aufbewahrt werden sollen. Jonny versuchte das Problem zunächst zu umgehen, indem bei Fehlern sofort die Korrektur anforderte. Dies ergibt aber einen wesentlich komplizierteren Protokollablauf, so dass die jetzige Implementierung doch wieder auf Protokolldateien zurückgreift.

5 In der Praxis

Ulf, DG1FAZ, berichtete mir, dass Paxon-Benutzer mit 76k8-Packet mittels DIDADIT erfolgreich Dateien von mehreren 100 Megabyte (!!) übertragen haben. Hier trat lediglich anfangs das Berechnen der MD5-Summe als Verzögerung zu Tage. Ansonsten hoffen wir darauf, dass möglichst bald BayBox-Versionen mit aktiviertem DIDADIT installiert sein werden.

6 Zukunft

Zur letztjährigen Packetradio Tagung hatten wir in einem nächtlichen Hack "TalKT" auf die Beine gestellt, welches - auf dem selben Codec basierend - ein Linux-Äquivalent zu FlexTalk sein sollte. Wie das so ist, ist die Sache danach im Sande verlaufen und aus heutiger Sicht gefällt mir die Implementierung auch nicht mehr. Die Audiodaten wurden zunächst eingelesen, nach Loslassen der "PTT-Taste" dann der codec aufgerufen, nach Beendigung des Codiervorganges wurden die Daten dann gesendet. In diesem Zusammenhang hatte ich ein HTTP-Ähnliches Protokoll eingesetzt, welches allerdings bereits zu Beginn der Übertragung die Länge der Daten benötigt. Dies ist aber nicht wünschenswert, wenn man parallel zur Spracheingabe codieren und übertragen will. Hier könnte eine einfache Variante von DIDADIT zum Einsatz kommen. Dank des Stuffings muss die Blockgröße nicht festgelegt sein, man kann Problemlos andere Blocktypen definieren, um zusätzliche Informationen in der Datenstrom einzustreuen, wie z.b. Stationsinformationen. Außerdem könnte ich mir vorstellen, einen multimedialen Convers Server zu implementieren.

Walter, DG9EP, hat versprochen einen ersten DIDADIT-Monitor zu implementieren. Dieser wird wohl zunächst nur unter Unix verfügbar sein.

7 Weiterführende Informationen

7.1 Im Internet:

1. Die DIDADIT-Spezifikation und weitere Informationen: http://1409.org/projects/didadit/
2. Paxon: http://www.paxon.de
3. WinSTOP: http://www.winstop.de
4. BayBox: http://www.baycom.org/baybox/

7.2 Im HamWeb

1. DIDADIT/LinKT: http://db0ais/~dg4iad/1409/
2. Paxon: http://db0ais/~paxon/