ProNET V3.0

User Documentation
23rd August, 1996

Michael Krause

This document was written in Texinfo and typeset using PasTEX and a modified `texinfo.tex` to provide nicer headings.

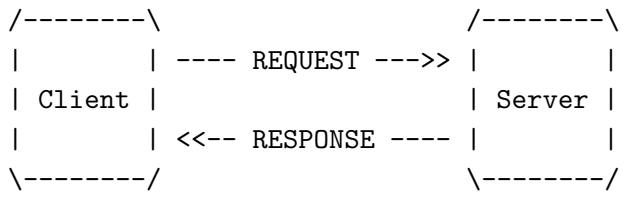
1 Introduction

1.1 What is *ProNET*?

ProNET is a simple network system. The most important use is to share devices like hard disks, floppies and CD-ROM drives between computers in the network.

I call it simple because it is not, will never be and is not ought to be a real competitor to existing standards like TCP/IP. I'm not a networking specialist; I didn't study a single book on this subject, so one can't expect *ProNET* to be an absolutely professional product.

The *ProNET* software is based on two-sided connections, which means that you can not connect more than two computers on one interface level - each level has always got two ends. This is called the *Client-Server-Model*:



But since *ProNET* uses a modular driver concept, you can open up more than one connection from each computer, so that you could hook together more than two of them.

Besides that, *ProNET* is limited to the Amiga series of computers. There will never be ports to other platforms, because the File System heavily relies on the Amiga way of handling data storage media.

Considering these facts, *ProNET* is the right program package for you, if you

1. don't want to spend much money
2. want perfect fit into your working environment
3. are mainly interested in connecting two Amigas.

This description might call the *ParNet* package into your mind, and indeed *ProNET* is ought to be the follow-up to that nice, but quite old and restricted software. These are some examples of improvements to Matt Dillon's work:

- modular driver concept, parallel, serial and MultiFaceCard3 transfer modules included
- installs a new AmigaDOS device for each target (no NET:)
- thus recognizes disk changes
- supports most of the V40 DosPackets
- 'cd's into network directories do work
- written in pure Assembler, thus short and a bit faster

Please do also note that there is no installation script – if you want to use *ProNET*, you should be familiar with the Shell and Text Editors.

1.2 System Requirements

The software generally requires at least Kickstart 2.0 and runs on any processor, with some exceptions: '`pronet-server`', '`pronet.device`', '`pronet-page`', '`pronet-run`', '`pronet-talk`' and the interface drivers do also work on older versions. Only Kickstart 1.3 has been tested, though.

2 Installation

Before we start moving around files, just let me explain what *Devices* are. If you find yourself familiar with these things you can skip this section and start installing. If you don't quite understand the install procedure, this section might enlighten you!

Because AmigaDOS can handle more than one data storage medium, each one has got its own name. The most known examples are '`DF0:`' for the internal floppy disk drive and '`HD0:`' for the boot partition of your hard disk. Every time you access a file on one of these media, you will state this name, provided the disk is not already the root of the current directory.

All these names describe a thing in AmigaDOS which is called the *AmigaDOS Device*. '`CD0:`', '`HD7:`', '`PRT:`' and '`RAM:`' all describe different AmigaDOS Devices, also simply called *Devices*. As you have probably noticed about the example of '`RAM:`', the AmigaDOS Device is only a *symbol* for a storage medium, since the Ram Disk is not really part of the hardware!

This device is not to be confused with the *Exec Device*, though both are just called *Device* in general. The former is a symbol, the latter is a program located in the ROM or in the '`DEVS:`' directory, the name of which is usually ending with '`.device`'. Exec devices are low-level drivers for hardware like hard disk drives and floppies. In this case, you don't need to know how they work or what they are good for.

2.1 Installing '`pronet.device`'

The `pronet.device` is the core of the whole *ProNET* package. It contains all routines for managing the data transfer between Amigas. It does *not* contain any transfer routines as was the case in version 1, because it turned out to be extremely hard to add new routines for other interfaces this way.

2.1.1 Copying the files

The device should be copied into the '`DEVS:`' directory of all participating machines. As soon as you receive your key file, copy it as '`pronet.key`' to '`DEVS:`'.

The *transfer routines* are external code modules stored in the '`DEVS:ProNET`' directory. I would recommend copying all of them, though you can later on delete the unused ones. Please don't copy them directly into '`DEVS:`', but keep them in their own directory!

Currently available transfer modules are '`internal-parallel`' for the internal parallel port (which, by the way, was the only possibility in *ProNET* V1), '`mfc-parallel`' for the parallel port of a MultiFaceCard3, and '`serial`' for serial ports which are controlled through a `serial.device` compatible Exec Device, that is '`BaudBandit.device`', '`duart.device`' etc.

2.1.2 Installing the drivers

Now create the file (or copy the example) '`DEVS:ProNET.config`'. This file will contain configuration data for the ports you are using for the network.

This file should contain at least one line of the form:

```
pronet-device-[unitnumber]: [drivername] [driverdata]
```

Each line describes a so called *Unit* of the device, representing a port accessable by one of the *ProNET* drivers. The more ports and drivers you have, the more Units you can create. You will be able to use every unit to connect to another machine: If your central Amiga contained a Multiport expansion containing 3 serial ports, you would be able to connect to 5 (five!) other Amigas by using the internal parallel and serial ports, as well as the Multiport card.

Units are identified by a positive number and defined by lines of the type shown above. `[drivername]` is replaced by the name of the external driver module, and `[driverdata]` is configuration data required by this driver.

2.1.2.1 '`internal-parallel`'

First we will install the internal-parallel driver as Unit number zero. Of course you can use a different Unit number, it's your choice – you don't have to use this driver at all! However, let's assume you want to use it: then we create or change the already present line into:

```
pronet-device-0: internal-parallel 0 5
```

The number 5 is required by the driver module and describes the priority of the transfer process. If you don't know what I'm talking about, leave this number. Lower numbers, like 0, can prevent the driver from stealing most of the CPU time.

The number 0 is the so called *machine number* and is essential for the correct working of the network. Two machines connected via the internal- or mfc-parallel driver need to have different numbers here. One machine requires a '0', the other one a '1'. If you gave both sides the same number, *ProNET* might not work.

This driver works with a *ParNet* compatible cable at the built-in parallel port.

2.1.2.2 'mfc-parallel'

This driver can be useful if you've got the MultiFaceCard3 from BSC. The on-board serial ports should be driven with the serial driver described below, the parallel ports are used with this driver.

Please note that you can only use MultiFaceCards which contain the 6821 PIA from Motorola – it has only been tested with the MultiFaceCard3.

Let's get down to the installation with Unit number two:

```
pronet-device-2: mfc-parallel 0 1 5
```

`0` is the number of the MFC parallel port, in case you've got more than one card in your Amiga.

The number 1 is the so called *machine number* and is essential for the correct working of the network. Two machines connected via the mfc- or internal-parallel driver need to have different numbers here. One machine requires a '0', the other one a '1'. If you give both sides the same number, you run the risk of some dangerous network deadlocks (no, you don't destroy your hardware)!

The number 5 is required by the driver module and describes the priority of the transfer process. If you don't know what I'm talking about, leave this number. Lower numbers, like 0, can prevent the driver from stealing most of the CPU time.

This driver works with a *ParNet* compatible cable. It is, without problems, possible to connect two Amigas with a MFC at one side and a built-in parallel port on the other side. This can degrade performance a bit, though.

2.1.2.3 'serial'

We will use Unit number one here:

```
pronet-device-1: serial BaudBandit.device 0 19200
```

`BaudBandit.device` is the name of the `serial.device` you want to use, `0` is the appropriate unit, which would describe the number of the serial port on multiport cards. Please note, that `8n1.device` does not function on an A500 with this driver. If you want to be really sure, use the original `serial.device` here.

`19200` is the baud rate, which must be equal on two computers connected together. Try how high you can set it! 19200 is a very safe value which even works on unaccelerated A500 models. Typical higher settings are 38400, 57600, 64000, 76800, 115200. The result of selecting a too high value is that the network could suddenly stop working, especially when the CPU working hard. The system could even work okay for half an hour and then stop working.

Without further arguments, this driver requires a 7-wire null-modem cable to work. However, the time could come when you only have 3 wires (e.g. when networking with a CD32). In this case, add the argument `3WIRE` to the end of the line (on both sides of the cable, of course). Note that 3-wire connections are generally less stable than 7-wire ones, especially under heavy multitasking.

2.2 Making the right cables

Please note that whatever you do, you'll do it on your own risk! I can't be held responsible for any damages you might do to your hardware.

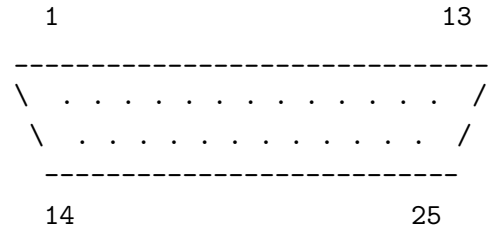
2.2.1 The *ParNet* cable

The following description is based on the *ParNet* documentation.

You are making a cable that connects the two **parallel** ports of your Amigas together. Connect D7-D0, SEL, POUT, and BUSY across; connect ACK to SEL locally:

```
(2-9)   D7-D0   ------------    D7-D0
(12)    POUT    ------------    POUT
(11)    BUSY    ------------    BUSY       PARALLEL PORT
(13)    SEL     --+------+--    SEL
(10)    ACK     -/        \-    ACK
(18-22) GND     ------------    GND  (18-22)
```

This is the DB-25 parallel connector seen from outside the computer:



Attention A1000 users: The parallel port is male, so you need a female connector! Other Amigas need a male connector.

The easiest thing to do is to buy a premade cable with all 25 lines passed and DB25 connectors on both ends (Double check the gender's that they match before you buy the cable!) And then rip it apart and cut-and-seal those wires which are not supposed to be connected. You also need to bridge ACK to SEL as per the diagram above *on both ends of the cable*, as shown above.

Double check the gender for the DB25 connectors you will need to connect to your Parallel Port. **Never plug in an unmodified cable between the two computers! Double check your cable before installation!**

2.2.1.1 Interference with built-in serial port

Warning: The RI (Ring Indicate) line on the Amiga's built-in serial port (Pin 22) uses the SEL line to source a transitor. This interferes with the SEL line which, as you can see, is part of the network.

Be sure that

- either no serial cable is attached or that
- It doesn't connect RI or that
- your modem doesn't connect RI internally or that
- you won't be called `;-)`

Otherwise, when receiving a signal on the RI line, the `internal-parallel` driver would simply crash!

2.2.2 The null-modem cable

2.2.2.1 7-wire schematics

Two ports should be connected in this way to give a 7-wire null-modem cable:

```
Shield Earth -> Shield Earth
System Earth -> System Earth
     TXD     ->      RXD
     RXD     ->      TXD
     RTS     ->      CTS
     CTS     ->      RTS
     DSR     ->      DTR
     DTR     ->      DSR
```

Do **not** connect Shield and System earth.

2.2.2.2 3-wire schematics

To get a 3-wire cable, leave out the DSR/DTR and RTS/CTS connections, so that we use the following schematics:

```
Shield Earth -> Shield Earth
System Earth -> System Earth
     TXD     ->      RXD
     RXD     ->      TXD
```

Do **not** connect Shield and System earth here either.

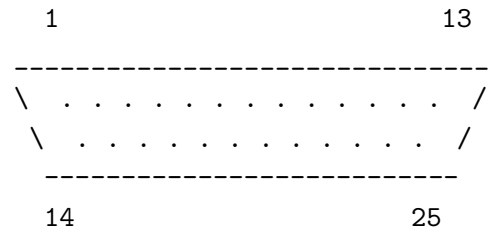
If you've got standard serial ports, please think about using 7-wire cables, as they are more reliable.

2.2.2.3 25-pin serial connectors

...are the standard Amiga ports.

```
Shield Earth (GND) - Pin 1
System Earth (GND) - Pin 7
     TXD           - Pin 2
     RXD           - Pin 3
     RTS           - Pin 4
     CTS           - Pin 5
     DSR           - Pin 6
     DTR           - Pin 20
```

This is the DB-25 RS-232 connector seen from outside the computer:

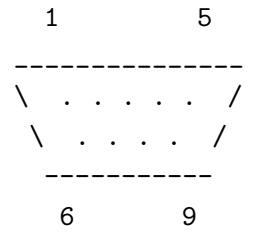


2.2.2.4 9-pin serial connectors

...are mostly used in the rest of the computer world and sometimes can even be found on Amiga multi-I/O cards such as the *MultiFaceCard3*.

```
Shield Earth (GND) - (connect to frame)
System Earth (GND) - Pin 5
     TXD           - Pin 3
     RXD           - Pin 2
     RTS           - Pin 7
     CTS           - Pin 8
     DSR           - Pin 6
     DTR           - Pin 4
```

This is the DB-9 RS-232 connector seen from outside the computer:



2.3 Installing the File System

This is easy: Let's assume your main working place is an A4000. You want to import the CD-ROM drive of a CDTV that is placed under your table into the A4000 so that you don't need to spend money on a new SCSI CD-ROM drive for your workstation. In this configuration it's quite obvious that the CDTV will be the network *Server*, whereas the A4000 will be the *Client*.

Just copy '`pronet-server`' into the '`C:`' directory or somewhere else in the path of the Server, that is the machine from which you want to import AmigaDOS Devices into the main machine. Then put the '`pronet-handler`' into the '`L:`' directory of the main machine, the Client.

That's it!

2.4 Installing the Utilities

Copy '`pronet-run`', '`pronet-page`', '`pronet-start`', '`pronet-stop`' and '`pronet-talk`' into the '`C:`' directory of your machines. That's it, again!

3 Usage

3.1 Starting the Server ('pronet-server')

For any Amiga you want to use as a server, you first have to start '`pronet-server`'. By default, this program waits for incoming request on *ProNET* Unit 0, but you can change this by supplying another value in the command line, so, if you had several connections to a central server computer, you would have to start '`pronet-server`' once for every Unit.

If you forget to start the server, client programs will simply go into an endless loop, so when later on nothing seems to work, always check if the server is running!

The server can be stopped at any time by issueing a '`CTRL-C`' signal. I would do so before rebooting the Client machine, since the server will then give notice of its departure. I suggest starting the server with `run <>nil:` so it doesn't block your shell. If you want to stop it, use the '`status`' and '`break`' commands.

3.2 Importing Devices ('pronet-start')

Starting the network is a very simple job, which is done by the '`pronet-start`' utility.

The command template is as follows:

```
pronet-start LOCALNAME/A,REMOTENAME/A,UNIT/N,FLAGS/N,UNIQUE/S
```

`LOCALNAME` is the name of the Device to create while `REMOTENAME` is the one you want to import from a remote Amiga. It is important not to give a volume name (e.g. 'Work') here but the device name (such as 'hd0')! Please also note that you must not append a ':' to the Device names!

As for the example from above, we could issue the command

```
pronet-start cd1 cd0
             ^^^
             This is what we want to
             call the device on the A4000.
                 ^^^
                 This is what the device is
                 called on the CDTV.
```

The rest of the arguments don't necessarily have to be specified:

`UNIT` is the *ProNET* unit you want to use (default 0). `FLAGS` currently has no effect (defaults to 0, too) and the `UNIQUE` switch makes Volume names unique. Imagine the following configuration:

Two Amigas with both a 'Workbench' partition are connected and Amiga 1 wants to access the Workbench partition of Amiga 2. You call '`pronet-start`' and what happens? Now you have got two 'Workbench' volume icons on your screen. This will confuse AmigaDOS quite heavily, which is to be avoided by using the `UNIQUE` switch. This will simply append a number to the imported volume name, corresponding to the *ProNET* unit you have used.

'`pronet-start`' returns an error message if something went wrong, that is e.g. when the remote device doesn't exist or the *ProNET* unit can't be found. Otherwise it immediately returns without further output.

Now repeat this step for every new device. Good luck!

3.3 '`pronet-stop`'

When you feel that you don't need a remote device any more, you can remove it with this command. Supply '`pronet-server`' with the device name you want to remove and it will do its job.

Why should you do this? Well, if you switched off the Server or reset it without stopping the Client, strange things might happen on the Client: You couldn't access any data any more, perhaps you couldn't even move your mouse – this is because the Client thinks the Server's still alive!

Beyond question: If you're going to switch off both computers anyway, you can skip this of course.

3.4 'pronet-flush'

You can use this command without further arguments to remove '`pronet.device`' from memory. This will free all resources allocated by the *ProNET* drivers. '`pronet-server`' and '`pronet-stop`' have this feature built-in. You will hardly find appropriate use for this command, but it's there for reasons of completeness anyway.

3.5 'pronet-page'

Using this program, you can send messages to users on remote machines. This works by calling '`pronet-page`' with the *ProNET* Unit and the message to be displayed on the machine connected to that Unit.

Example:

```
pronet-page 0 Hello there!
```

The message will be displayed in a Guru-like alert window at the top of the screen.

3.6 'pronet-run'

By using this program, you can run programs on a networked machine without having to touch its keyboard or mouse. Call it with the *ProNET* Unit and the command to be run on the corresponding Amiga. The command string *must* be put in "s.

Example:

```
pronet-run 0 "dir hd0: all"
```

3.7 'pronet-talk'

This program can be used to chat between two machines. All you need to do is call '`pronet-talk`' on *both* machines with the right *ProNET* Unit as an argument. By default, this program uses Unit 0.

4 Advanced Usage

In this chapter, I want to give you some hints to how to use the full power of ProNET. There are some things which are not very obvious and may be useful for some of you.

4.1 Reconnection

Okay, this is not quite what I would call advanced usage, but it's the right place for this anyway.

Let's get back to the example from the *Importing Devices* section. You will sometimes have to reboot your A4000, because some program has crashed (uuh not really, we use an Amiga ;-)). When you want to start the network again, you don't have to reboot the CDTV, too – It's completely okay to issue new '`pronet-start`' commands after a Client reboot without rebooting the Server. That's all.

4.2 Networking in both directions

Most configurations will look similar to the one described above, i.e. one main machine, the working place, and a server which is no working place. However, if you are networking with a friend's system and you are both working on the machines, it's no problem to install the network in both directions, that is both machines are Server and Client at the same time. The two can each access the other one's file systems.

4.3 A network over the phone line

Imagine you have a friend living far away from you, and you want to help him install some new programs. You could call him and describe exactly what he has to do, but this way you will spend far too much time, because he will not understand (This presumption has been derived from Murphy's Law and proved to be fact in many situations `;-)`).

You could also make a modem connection, install a network between your computers and copy all the files to the right places, make some shell scripts etc. on your own. Does this sound interesting? Then read on!

4.3.1 Preparation

Both sides must define a ProNET Unit using the '`serial`' driver. The respective port will not have a null-modem cable connected to it, but a real modem! As usual, set the serial port speed to four times the modem speed, if your port does it. The speed settings on both sides don't have to be equal, because the modem manages the transfer speed on its own.

Both modems must be configured to ignore the DTR (data terminal ready) signal, which is in most cases done by issueing the command `AT&D0`.

4.3.2 How to connect

Load your favourite terminal programs now. Decide one of you to be the caller, who then dials the other one's number via `ATD`. As soon as he has answered with `ATA` and you've read the `CONNECT` message, both sides quit the terminal.

Now simply proceed as if you were connected with a simple null-modem cable: Start the server programs and install the devices (of course directed to the right unit!). That's it. Not very fast, but nice anyway `;-)`

5 Internals

5.1 How to create *ProNET* drivers

A *ProNET* driver is an ordinary executable residing in the '`DEVS:ProNET/`' directory. This executable will be loaded for every *ProNET* Unit requiring it; it doesn't have to be re-entrant. The code is entered at the very first position, which must contain an **initialization routine**:

d0 contains the bytes `"RST!"` for identification, so that this driver can not be executed by accident.

d1.l is a Flags register. It is **not** mapped to the Flags argument of the OpenDevice() call!! For Version 3, this contains a `1`; if it contains another value, your driver should return an error.

a1 contains a pointer to the configuration string following the driver ID. Example:

```
pronet-device-7: yourdriver bla this is data for your driver
```

In this case, a1 would contain a pointer to the null-terminated string '`bla this is data for your driver`'.

a0 finally points to an array which must be filled out by your routine:

```
UBYTE ReadSignalBit
UBYTE pad0
APTR ReadQuery
APTR Read
APTR Write
APTR    Exit
```

ReadSignalBit must contain a valid signal bit for the '`pronet.device`' task. The task will include this bit into its Wait() loop and call the function pointed to by `ReadQuery`, if this bit becomes set.

Your **ReadQuery** routine must analyze the incoming packet and return the following values:

```
d0.w length of the packet's main data
d1.w ProNET destination port
d2.w ProNET source port
```

If your routine comes to the conclusion that it was called by mistake and no incoming packet is pending, it must return NULL in d0! If this routine returns something useful, it's guaranteed that the routine pointed to by `Read` will be called after that.

The **Read** routine will be called with a pointer to the place to put the packet data into in a0. It must copy the incoming packet there and return. You don't have to return an error code, because this routine must **always** work!

Let's come to the **Write** routine now. It is called when the '`pronet.device`' wants your driver to transmit data to the other machine. The routine will be called with following arguments:

```
a0 *data1
d0.l length of data1 (guaranteed to be even)
a1 *data2
d1.l length of data2 (guaranteed to be even)
d2 destination port
d3 source port
```

Both chunks of data must be appended so that the other side thinks you've transmitted only one big chunk of data! The expression d0+d1 is always guaranteed to be lower than or equal to $4000. If d1 contains zero, only one chunk of data is transmitted. If your routine succeeds in transmitting the data, you must return NULL in d0; if timeout occured or the line was busy or anything else which prevented you from transmitting, you must return a different value.

The **Exit** routine is new for ProNET V3, since this version offers network shutdown. Do here what is necessary, in your opinion ;-)

The initialization routine must return an error code in d0. If NULL is returned, the initialization succeeded, other values indicate an error.

All routines must preserve d2-d7/a2-a6.

5.2 Supported DosPacket Types

ProNET supports most of the new DosPackets introduced with V36 and V39: Perhaps you've sometimes seen one of these '2.0 Pkt ACT_PARENT_FH' requesters from *ParNet* - they are no longer annoying you! This is a list of all packet types supported by *ProNET*:

```
ACTION_FINDINPUT
ACTION_FINDUPDATE       V33
ACTION_FINDOUTPUT
ACTION_END
ACTION_READ
ACTION_WRITE
ACTION_SEEK
ACTION_CURRENT_VOLUME
ACTION_SET_FILE_SIZE    V36
ACTION_LOCK_RECORD      V36
ACTION_FREE_RECORD      V36

ACTION_LOCATE_OBJECT
ACTION_FREE_LOCK
ACTION_COPY_DIR
ACTION_PARENT
ACTION_SAME_LOCK        V36
ACTION_CREATE_DIR
ACTION_CHANGE_MODE      V36
ACTION_FH_FROM_LOCK     V36
ACTION_COPY_DIR_FH      V36
ACTION_PARENT_FH        V36
ACTION_EXAMINE_OBJECT
ACTION_EXAMINE_NEXT
ACTION_EXAMINE_FH       V36

ACTION_DELETE_OBJECT
ACTION_RENAME_OBJECT
ACTION_MAKE_LINK        V36
ACTION_READ_LINK        V36
ACTION_SET_COMMENT
ACTION_SET_DATE
ACTION_SET_PROTECT
ACTION_INFO
ACTION_RENAME_DISK
ACTION_INHIBIT
ACTION_FORMAT           V36
ACTION_SERIALIZE_DISK   V39
ACTION_MORE_CACHE
ACTION_WRITE_PROTECT
ACTION_IS_FILESYSTEM    V36

ACTION_NIL
```

```
ACTION_FLUSH
ACTION_DISK_INFO
```

These packets are not yet supported and might be implemented later:

```
ACTION_EXAMINE_ALL      V36
ACTION_EXAMINE_ALL_END  V39
ACTION_SET_OWNER        V39
ACTION_ADD_NOTIFY       V36
ACTION_REMOVE_NOTIFY    V36
```

6 Other Topics

6.1 Registration

ProNET is Shareware. You might have noticed that without a key file, the network stops functioning after 5 minutes. To obtain a personalized key file, please send 20DM or US$ 15 (no coins please) to my address (see Support section). Users that have registered for Version 2 can update by sending 10DM.

Please write down your complete address in a **readable** manner. If you've got an e-mail address, I can also send the key file uuencoded to you. Please also clearly state which version of *ProNET* you are using.

When I receive a registration, I will generally answer it within two or three days - if you have to wait three weeks, you can be sure I was on vacation. Together with the key file, you will receive the latest version of *ProNET*.

6.2 Support

Due to the low registration fee and the costs of ordinary mail, it's generally not possible for me to answer every question concerning *ProNET* by snail-mail. If you've got an Internet address, the chance of an answer will rapidly increase. Also please do try to find a solution to your problem in the FAQ section of this guide first.

New versions of *ProNET* will be immediately spread via Aminet and local BBSes. You can also check out the World Wide Web support site at

```
http://www.szczecin.pl/~rawstyle/pronet.html
```

If you want to report a bug, contact me at:

Michael Krause
Mannesallee 24
D-21107 Hamburg
Germany

rawstyle@blackbox.dame.de

Bug reports should be as exact concerning the configuration as possible; but please don't write which mouse you have got ;-)

6.3 Credits & Thanks

All *ProNET* software by

Michael Krause

ParNet (inspiration) and design of the parallel connection cable by

Matt Dillon

Beta Testing by (thank you very much!)

Thomas Schwarz, **Wolfgang Gutberlet**, **Anti-"It-does-not-work"-brain**

Many thanks to Nik Soggia <nsoggia@falcon.telnetwork.it> for finding a solution to the ReqTools problem from V1!

Besides all that I also want to thank all people that sent letters containing more than just '`ProNET does not work`' and all the ones encouraging me to continue working on this package. This update would never have been released if I hadn't received so much mail from you!

6.4 History

6.4.1 Version 3

released on 23-Aug-96 including pronet.device **36.6**, pronet-handler **36.4**, pronet-server **36.4**, pronet-start **36.3**, pronet-stop **36.1**, pronet-flush **36.0**, pronet-talk **36.0**, pronet-run **36.1**, pronet-page **36.0**.

- No more MountList and ProNET.config fiddling, '`pronet-start`' does everything now! Thus, '`pronet-handler`' had to be changed, too.
- Wrote a MultiFaceCard3 driver.
- Debugged the serial driver.
- Network can be shut down without having to reset.
- Completely new documentation.
- The '`serial`' driver can be advised to use 3-wire cables now.
- Added commandline checkings to '`pronet-page`' and '`pronet-run`'.
- '`pronet-talk`' uses fixed port 1994.
- New '`pronet.device`' error code '`PNDERR_UNIT_NOT_DEFINED`', fixed unit open failure handling.

6.4.2 Version 2.1

(bugfix) released on 08-Jul-95 including pronet-talk **34.2**, pronet-server **35.0**.

- pronet-talk AND pronet-server didn't accept the UNIT argument...

6.4.3 Version 2

released on 24-Jun-95 including pronet.device **35.4**, pronet-handler **34.2**, pronet-server **34.9**, pronet-talk **34.1**, pronet-run **34.1**, pronet-page **34.0**.

- If the main machine resets, there is no need to reset the server any more.
- The device doesn't freeze the machine.
- Priority of transfer process can be set now
- Disk change recognition improved again
- Completely new documentation

- READ-actions are now transferred splitted, and don't require as much RAM on the server as in V1.
- EXAMINE-Packets could cause memory loss on the server
- Completely new pronet.device, with modular driver concept
- "+"-Expansion >> Unitnumber
- C include file
- everything I forgot to write here
- Lots of bug fixes, thus a lot more stable

6.4.4 Version 1

released on 01-Nov-94 (initial release) including pronet.device **33.9**, pronet-handler **33.5**, pronet-server **33.5**, pronet-talk **33.0**, pronet-run **33.0**, pronet-page **33.0**.

6.4.5 Version 0

I got the idea for *ProNET* in December 1993 after being disappointed by ParNet and started programming the device in April 1994.

6.5 Future

- Making the server rebootable!
- Finishing the MUI-GUI.
- Make volume name extension freely configurable.
- Local caching of :.backdrop and :Disk.info
- Net-Keyboard, Net-Mouse
- XPK crunched packet transfer (a la v42)
- Security: Make devices unavailable to the network
- Notification DOS packets.
- Fixing the DiskSalv problem.
- Net-CON:
- Net-Serial-Device
- ProNET-Driver for SANA-II-Devices.

- ProNET-Driver for parnet.device
- ProNET-Driver for AmigaDOS-Devices (PAR: etc.)
- SANA-II-Device for pronet.device

7 FAQ - Frequently Asked Questions

- Is it possible to stop the network in order to do some printing and then reconnect both Amigas via a parallel port switch?

  No, this is not possible, and it will not be in future, because the network relies on the parallel port registers not to be changed by the Operating System.

- Is there a possibility to start a program on the remote machine and see its output in a CON:-Window on the main machine?

  No, this is not possible as yet, but will perhaps be implemented in the future.

- Is it possible to use the serial port of the remote machine to e.g. connect a second modem or using a MIDI interface and a modem at the same time?

  This is a thing which is on my to-do list. Thanks to Giovanni Gigante for the nice idea!

- Will the 'pronet-handler' ever run on Kickstarts earlier than 2.0?

  No.

- Would a special 020/030/040/060 version of the device or the drivers improve *ProNET*'s speed?

  No.

- I've discovered a bug: When I reset the main computer and mount all network devices again, I can continue working without problems. However, when I reset the machine on which pronet-server is running and start pronet-server again, the network is dead.

  This is no bug, because the pronet-server creates some structures for the main machine which are cleared if you reset the server machine! A conclusion from that is, that 'crossed-over' networks, where both machines are server and client, are not allowed to do a reset at all! This is an essential difference between *ProNET* and professional network packages.

- Can pronet.device used as an interface driver for SANA-II compliant network software?

  No, this is not possible, since my device is not SANA-II compliant :) However, I intend to code a device stub so you could use all ProNET drivers with e.g. Envoy or AmiTCP.

- Why didn't I program my device SANA-II compliant??

  This is because when I started programming *ProNET*, I didn't know that this standard existed.

- Do I have to call '`pronet-start`' for every drive or can I somehow combine calls for more than one device?

  No. Create a shell script if you are too lazy ;-)

- I use a CDTV / A500+A570 as a server. The CD-ROM drive seems not to recognize disk changes. Is it a bug in *ProNET*?

  No, it's a bug in the CDTV file system. Simply issue a `diskchange cd0:` command on the client or server machine.

- I use the Workbench to copy large files to the server. Sometimes the computer says 'not enough memory'.

  Try to use the Shell copy command or a DOpus like tool instead. Explanation for programmers: Large ACTION_WRITEs lead to problems, since a corresponding amount of memory needs to be allocated on the server machine. Writing step-by-step is too dangerous in disk-nearly-full situations.
- I'm using DiskSalv to salvage a disk via ProNET to a remote Amiga. This is awfully slow, however.

  This is a problem I didn't found a solution to yet, but I'm working on it.
- Creating new drawers via ProNET takes some seconds, too.

  This seems to be the same problem as the last one.

Hmm... sorry that there were not too much questions I could answer with a simple 'Yes' :) ohh well but let me think about that again... ahhh here it is:

- Is *ProNET* better than ParNet?

  <Guess> :-)

8 Legal stuff

8.1 Disclaimer

"I won't read this yet again..."

8.2 License

- You may not copy your personal key file to other persons nor make it publically available. You may not use other key files than your very own with your *ProNET* copy. You may only use this key file on your very own computers.
- Except when otherwise stated in this documentation, you may not copy and/or distribute this program without the accompanying documentation and other additional files that came with the original. You may not copy and/or distribute modified versions of this program.
- You may not copy, modify, sublicense, distribute or transfer the program except as expressly provided under this license. Any attempt otherwise to copy, modify, sublicense, distribute or transfer the program is void, and will automatically terminate your rights to use the program under this license. However, parties who have received copies, or rights to use copies, from you under this license will not have their licenses terminated so long as such parties remain in full compliance.
- By copying, distributing and/or using the program you indicate your acceptance of this license to do so, and all its terms and conditions.
- Each time you redistribute the program, the recipient automatically receives a license from the original licensor to copy, distribute and/or use the program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein.
- You agree to cease distributing the program and data involved if requested to do so by the author.

Table of Contents