USER DOCUMENTATION – A2

[A2-Ud]

Contents

1 Introduction

This document describes the interface I_CL, which is used towards the package `no.ntnu.fp.net.cl`, also known as A2. In addition, a description of the errors that can be expected from A2 is given, as well as an example on how to use A2.

2 Interface descriptions

2.1 A2

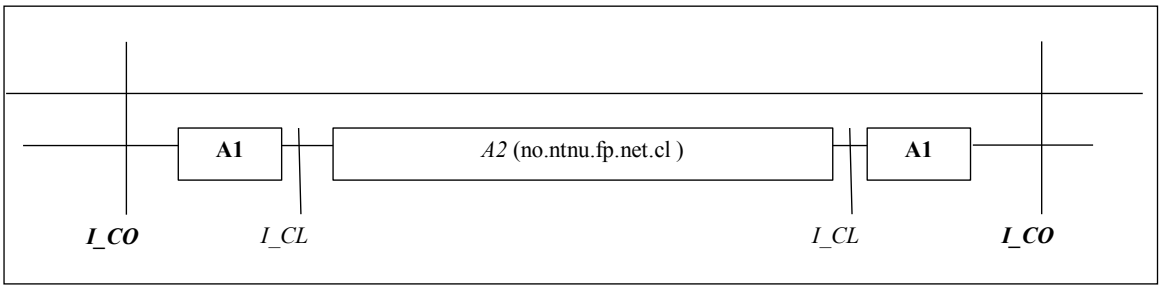

Figure 1: The overall picture

A2 provides a connectionless communication service. By connectionless, it is given that it does not guarantee the data sent has been delivered to the receiver. It does neither guarantee the data to be error free. The only thing that is guaranteed is that the data is attempted sent.

A2 is a custom made network connection. Therefore you have the opportunity to control how it shall behave. The behaviour is controlled by using a settings control panel, as documented in [Adm-Ud]. It is up to the user of the interface to a connection-oriented

service based upon the connectionless service given by A2. This would for example include creating connections for each data that shall be sent, and to guarantee the arrival of this data at the destination.

The `ClSocket` class contains the methods that I_Cl defines. The interaction with A2 is strictly defined to be to send data to a host or to receive data from another host. The method `cancelReceive()` is available to unblock the `ClSocket` object when it is blocked in `receive()`.

- Constructor: empty.
- `send(KtnDatagram inPacket)`- This method is used to transmit a `KtnDatagram` to the given host. The host to transmit to is given in the `KtnDatagram` object. This method will throw a `ClException` if the host address is not found. It can also throw an `IOException` if there is an error in the network for example.
- `receive(int port)`- This method blocks until a `KtnDatagram` is received on the port specified. When a datagram is received, it is returned to the caller of receive. This method throws a IOException if an error occurs.
- `cancelReceive()`- This method interrupts the object and makes it stop blocking and receiving.

3 Textual description

3.1 Errors

When sending data using `ClSocket` (A2) there are several errors that can occur. In [Table 1] the errors are summarized.

Name	**Cause**	**Consequence**
Package lost	The package did not manage to come through to the destination	The packet does not arrive at the destination and the information is lost.
Package delayed	The package got delayed somewhere but appears eventually after some delay.	The package may occur twice as the package may be retransmitted because it was thought to be lost.
Package has errors	The package has been contaminated somewhere along the way and is not valid any more. The checksum is wrong.	The package can contain the wrong information when arriving at the destination. If the header is altered, then it can end up at the wrong computer.
Ghost package	A package from nowhere or anywhere appears to belong	A packet that should not be received is received. Can

	to this computer and process, based on the header, and is therefore caught.	often be eliminated with checksum check.

Table 1: Errors that can occur in A2

3.2 How to use A2

A2 is a connectionless implementation and therefore to send data you just have to wrap the data in a `KtnDatagram` instance, and use for example `(new ClSocket()).send(instance of KtnDatagram)` to send the data to the destination specified in the datagram. To receive data one can use `(new ClSocket()).receive(portNr)`. One can of course also use the same instance for all communication. For an example see [Figure 2]. In the example, receiver's code and sender's code must run in separate threads to work. This is not a running example but merely an example of how the specific code could look.

```
/* Start of receiver's code */
ClSocket clSocket = new ClSocket();
KtnDatagram packet;

/* receiving the packet containing a string */
packet =clSocket.receive();
/* End of receiver's code */


/* Start of sender's code */
ClSocket clSocket = new ClSocket();
KtnDatagram packet = new KtnDatagram();

packet.setDest_addr("localhost");
packet.setDest_port(2222);
packet.setPayload("Hello World");

/* Sending the packet containing a string */
clSocket.send(packet);
/* End of sender's code */
```

Figure 2: Example of use

4 References

This document references to the following other documents:

[Adm-Ud] User documentation – Administration package