

Hierarchical Protocol

Abstract

This example shows how the CP-net from “Simple Protocol” can be turned into a hierarchical CP-net – with separate pages (subnets) for the *Sender*, the *Network* and the *Receiver* part. The protocol is modified to accommodate multiple *Receivers*.

Developed and Maintained by:

Kurt Jensen, Aarhus University, Denmark (kjensen@daimi.aau.dk).

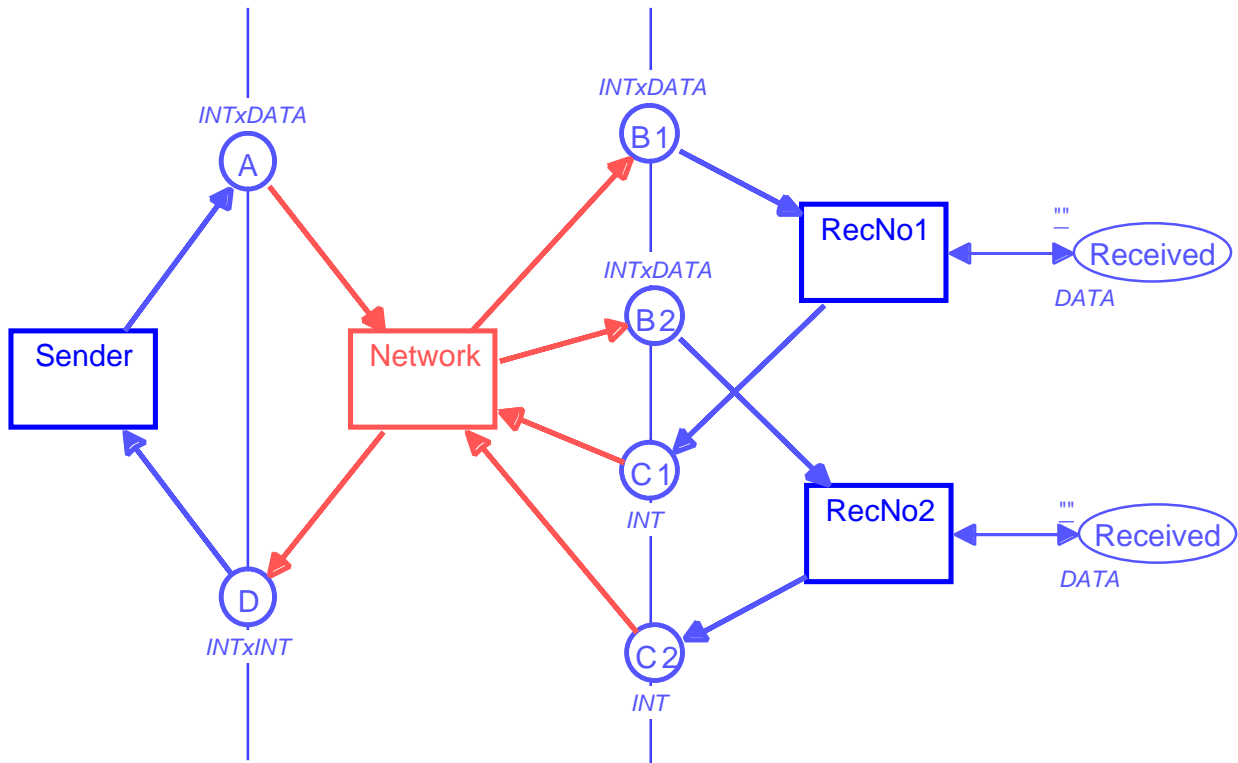
Graphical Quality

The figures in this document are inserted via PICT format. This is why some of the arcs and place borders look a bit ragged. A postscript printout from Design/CPN (and the screen image in Design/CPN) has much higher graphical quality.

CPN Model

The most abstract page looks as shown below. It tells us that we have a *Sender*, a *Network* and two *Receivers*. The basic idea is that the *Sender* sends messages which the *Network* broadcasts to the two *Receivers*. Analogously, the *Receivers* send acknowledgments which the *Network* transmits to the *Sender*.

Overview



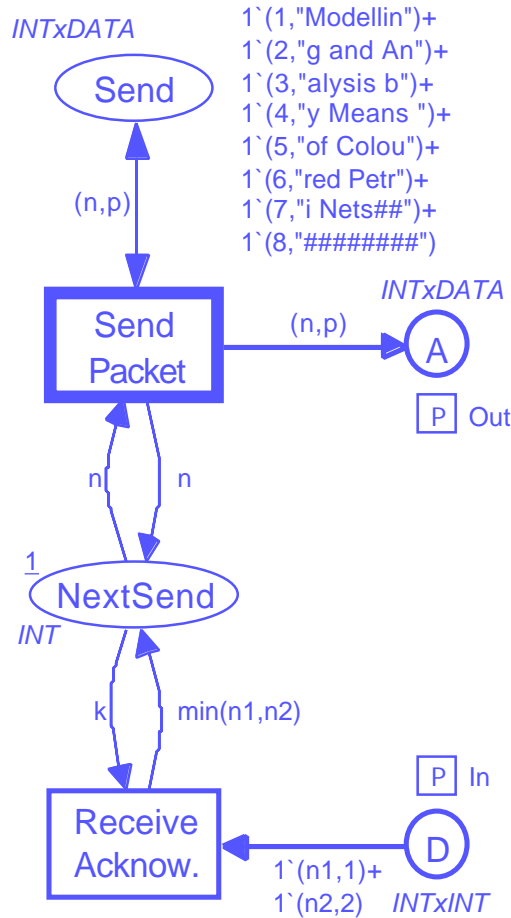
```

color INT = int;
color DATA = string;
color INTxDATA = product INT * DATA;
color INTxINT = product INT * INT;
var n, k, n1, n2 : INT;
var p, str : DATA;
val stop = "#####";

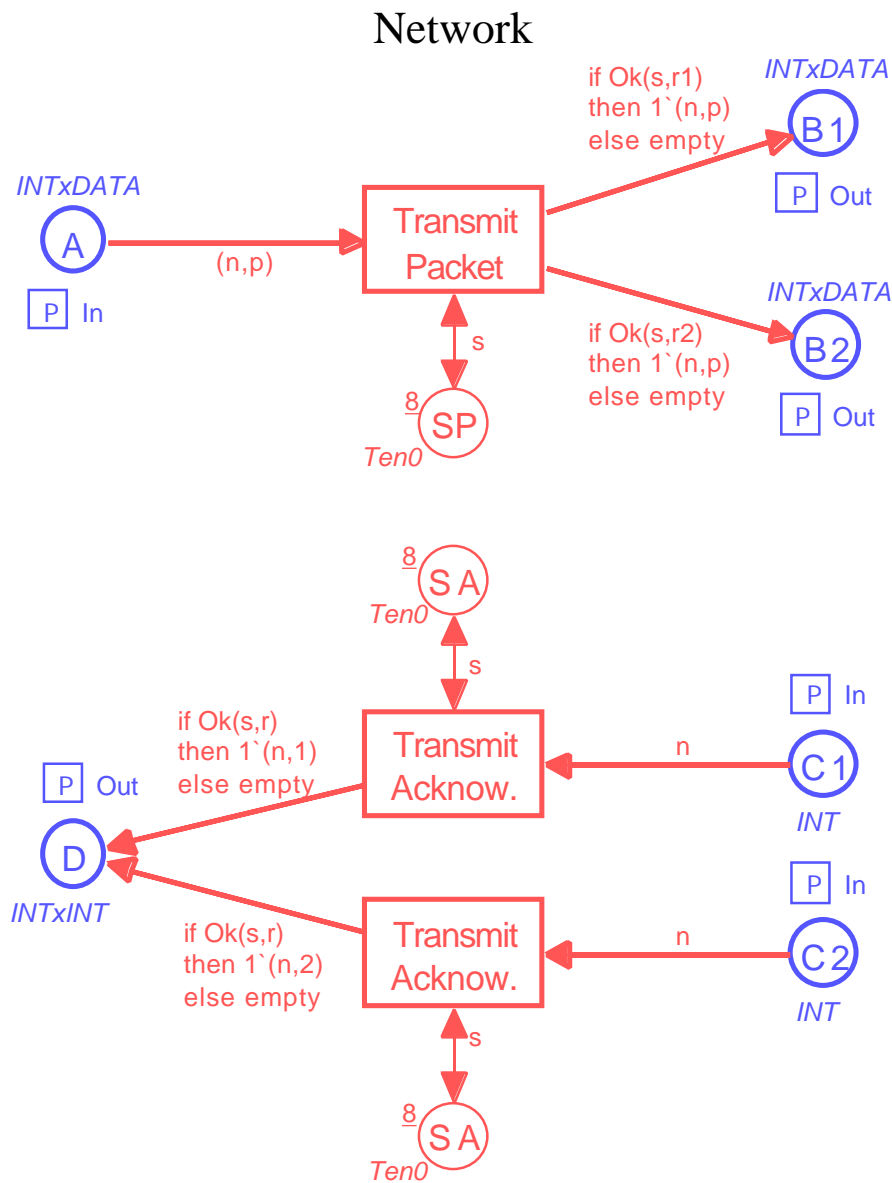
color Ten0 = int with 0..10;
color Ten1 = int with 1..10;
var s : Ten0; var r, r1, r2 : Ten1;
fun Ok(s:Ten0,r:Ten1) = (r<=s);
  
```

The *Sender* part is similar to the sender part of the “Simple Protocol”. The only difference is that *Receive Acknowledgment* now needs an acknowledgment from each of the two *Receivers* in order to become enabled. Each acknowledgment is a pair where the first element is the contents, while the second element indicates whether it came from *Receiver* one or two. Since packets are sent by means of broadcasts, the *Sender* needs to wait for the slowest of the two *Receivers* (or the most unlucky one). Hence *Next Send* is updated to be the minimum of the two acknowledgment values.

Sender



The *Network* part is similar to the network part of the “Simple Protocol”. However, again there are a few differences. *Transmit Packet* produces packets at two different output places *B1* and *B2*. The packets at *B1* are for the first *Receiver*, while the packets at *B2* are for the second. It should be noted that we use two different variables *r1* and *r2* to determine whether the packets for *B1* and *B2* are lost or not. This means that we model a broadcast in which one of the *Receivers* may get a packet while the other does not. If we replace *r1* and *r2* with a single common variable *r*, we get a broadcast where the two *Receivers* get exactly the same packets. *Transmit Acknowledgment* is split in two. The upper one handles acknowledgments from the first *Receiver*, while the lower one handles those from the second. Both of them modify the acknowledgment, by adding information telling the *Sender* where the acknowledgment came from.



The *Receiver* part is totally identical to the receiver part of the simple protocol. However, it should be noted that the *Receiver* page is used by two substitution transitions, *RecNo1* and *RecNo2*. This means that we will have two instances of the subnet – during a simulation. The two instances may have different markings and different enabling, otherwise they will be identical. Design/CPN displays one instance at a time. To see another instance, the user applies the **Switch Instance** command in the **Sim** menu.

Receiver

