Channels

Exercise 1 The perfect point-to-point links abstraction allows messages from one sender to arrive at a receiver in a different order than they were sent. Some applications rely on *first-in first-out* (FIFO) order communication, however. Specify a FIFO-order perfect point-to-point links abstraction which ensures, in addition to the guarantees of perfect point-to-point links, that messages are not reordered.

Solution A specification of FIFO-order perfect point-to-point links is shown in Module 1.

Exercise 2 Provide an implementation of FIFO-order perfect point-to-point links (*Exercise 1*) on top of perfect point-to-point links using sequence numbers.

Solution Algorithm 2 implements FIFO-order perfect point-to-point links on top of perfect point-to-point links.

Exercise 3 Does the following statement satisfy the synchronous-computation assumption? "On my server, no request ever takes more than one week to be processed."

Solution The answer is yes. This is because the time it takes for the server to process a request is bounded and known, one week.

Exercise 4 In a fail-stop model, which of the following properties are safety properties?

1. *Every process that crashes is eventually detected.* This is a liveness property; we can never exhibit a time t in some execution and state that the property is violated. There is always the hope that eventually the failure detector detects the crashes.
2. *No process is detected before it crashes.* This is a safety property. If a process is detected at time t before it has crashed, then the property is violated at time t.
3. *No two processes decide differently*. This is also a safety property, because it can be violated at some time t and never be satisfied again.
4. *No two correct processes decide differently.* Since a correct process is a process that never crashes and executes an infinite number of steps, the set of correct processes is known a priori. Therefore, this property is also a safety property: once two correct processes have decided differently in some partial execution, no matter how we extend the execution, the property would not be satisfied.
5. *Every correct process decides before t time units.* This is a safety property: it can be violated at some time during an execution, where all correct processes have executed t of their own steps. If violated at that time, there is no hope that it will be satisfied again.
6. *If some correct process decides, then every correct process decides.* This is a liveness property: there is always the hope that the property is satisfied. It is interesting to note that the property can actually be satisfied by having the processes not do anything. Hence, the intuition that a safety property is one that is satisfied by doing nothing may be misleading.

Module:
  Name: FIFOPerfectPointToPointLinks, **instance** *fpl*.
Events:
  Request: $\langle$ *fpl*, *Send* | *q*, *m* $\rangle$: Requests to send message *m* to process *q*.
  Indication: $\langle$ fpl, Deliver | p, m $\rangle$: Delivers message m sent by process p.
Properties:
  FPL1: *Reliable delivery*: If a correct process *p* sends a message *m* to a correct process *q*, then *q* eventually delivers *m*.
  FPL2: *No duplication*: No message is delivered by a process more than once.
  FPL3: *No creation*: If some process *q* delivers a message *m* with sender *p*, then *m* was previously sent to *q* by process *p*.
  FPL4: *FIFO delivery*: If some process sends message *m1* before it sends message *m2*, then no correct process delivers *m2* unless it has already delivered *m1*.

Figure 1. Interface and properties of FIFO-order perfect point-to-point links

```
Implements:
  FIFOPerfectPointToPointLinks, instance fpl.
Uses:
  PerfectPointToPointLinks, instance pl.

upon event ⟨ fpl, Init ⟩ do
   forall $p \in \Pi$ do
      lsn[p] := 0;
      next[p] := 1;

upon event ⟨ fpl, Send | q, m ⟩ do
  lsn[q] := lsn[q] + 1;
  trigger ⟨ pl, Send | q, (m, lsn[q]) ⟩;

upon event ⟨ pl, Deliver | p, (m, sn) ⟩ do
  pending := pending $\cup$ {(p, m, sn)};
   while exists (q, n, sn' ) $\in$ pending such that sn' = next[q] do
      next[q] := next[q] + 1;
      pending := pending \ {(q, n, sn' )};
      trigger ⟨ fpl, Deliver | q, n ⟩;
```

Figure 2. Implementation of FIFO-order perfect point-to-point links on top of perfect point-to-point links.