```
optype stream = (int);  # type of data stream operations

module Merge[i = 1 to n]
  op in1 stream, in2 stream;  # input streams
  op initialize(cap stream);  # link to output stream
body
  int v1, v2;    # input values from streams 1 and 2
  cap stream out;  # capability for output stream
  sem empty1 = 1, full1 = 0, empty2 = 1, full2 = 0;

  proc initialize(output) {  # provide output stream
    out = output;
  }

  proc in1(value1) {  # produce next value for stream 1
    P(empty1); v1 = value1; V(full1);
  }

  proc in2(value2) {  # produce next value for stream 2
    P(empty2); v2 = value2; V(full2);
  }

  process M {
    P(full1); P(full2);  # wait for two input values
    while (v1 != EOS and v2 != EOS)
      if (v1 <= v2)
          { call out(v1); V(empty1); P(full1); }
      else  # v2 < v1
          { call out(v2); V(empty2); P(full2); }
    # consume the rest of the non-empty input stream
    if (v1 == EOS)
      while (v2 != EOS)
          { call out(v2); V(empty2); P(full2); }
    else  # v2 == EOS
      while (v1 != EOS)
          { call out(v1); V(empty1); P(full1); }
    call out(EOS);  # append sentinel
  }
end Merge
```

Figure 8.3 Merge-sort filters using RPC.