```
upon event <Init> do
        forall Pi ∈ Π do
                delivered[Pi] := 0;
                missing[Pi] := 0;
        lsn := 0; stored := 0;

procedure gossip (msg) is
        forall t ∈ pick-targets (fanout) do
        trigger <flp2pSend | t, msg>;

upon event <pbBroadcast | m> do
        lsn := lsn+1; trigger <unBroadcast | [Data, self, m, lsn]>;

upon event < unDeliver | Pi, [DATA, Sm, m, SNm]> do
        if (store-thresold < random()) then
                stored := stored ∪ { [DATA, Sm, m, SNm] };
        if (SNm >= delivered[Sm] + 1)  then
                trigger < pbDeliver | Sm, m >;                    //deliver immediately
                forall seqnb ∈ [delivered[Sm] + 1, SNm - 1] do    //usually you use [n, n + |m|]
//notation
                        gossip ([REQUEST, self, Sm, seqnb, maxrounds – 1]);
                        missing[Pi] := missing[Pi] ∪  seqnb;
                delivered[Sm] := SNm;
                startTimer (TimeDelay, Pi, SNm);
        else if (SNm ∈ missing[Sm]) then
                missing[Sm] := missing[Sm] \ SNm;
                trigger < pbDeliver | Sm, m >;

upon event < flp2pDeliver | Pj, [REQUEST, Pi, Sm, SNm, r] > do
        if ([DATA, Sm, m, SNm] ∈ stored) then
                trigger < flp2pSend | Pi, [DATA, Sm, m, SNm] >;
        else if (r > 0) then
                gossip ([REQUEST, Pi, Sm, SNm, r – 1]);

upon event < flp2pDeliver | Pj, [DATA, Sm, m, SNm]> do
        if (SNm ∈ missing[Sm]) then
                missing[Sm] := missing[Sm] \ SNm;
                trigger < pbDeliver | Sm, m >;
```