```
function P_k.BufferData
    ProcessMsg
    chunk_to_play = x
    while x < chunk_to_play or (x − chunk_to_play) < B/2 do
        ProcessMsg()
function P_k.ProcessMsg
    if x ≥ 0 then
        ProcessChunkMsg
    else
        ProcessControlMsg
function P_k.ProcessChunkMsg
    if buffer[x%B].chunk_number = x then
        ProcessDupplicateChunk
    else
        ProcessNewChunk
function P_k.ProcessDuplicateChunk
    [prune x] → sender
function P_k.ProcessNewChunk
    buffer[x%B] = [x, origin, chunk]
    if sender ≠ S then
        debt[sender] = debt[sender] − 1
        forward[P_k] = forward[P_k] ∪ sender
    for all P_i ∈ forward[origin] do
        pending[P_i] = pending[P_i] ∪ x
    for all chunk_number ∈ pending[neighbor] do
        buffer[x%B] → neighbor
        pending[neighbor] = pending[neighbor] \ x
        debt[neighbor] = debt[neighbor] + 1
        if debt[neighbor] > D then
            for all peers_list ∈ forward do
                peers_list = peers_list \ neighbor
    neighbor =
function P_k.ProcessControlMsg
    if x = request then
        origin = buffer[chunk%B].origin
        forward[origin] = forward[origin] ∪ sender
    if x = prune then
        origin = buffer[chunk%B].origin
        forward[origin] = forward[origin] \ sender
    if x = hello then
        Peer.add_neighbor(sender)
    if x = goodbye then
        for all peers_list ∈ forward do
```

peers_list = peers_list \ sende