```
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 $x \geq 0$ then
        ProcessChunkMsg
    else
        ProcessControlMsg
function $P_k$.ProcessChunkMsg
    if buffer$[x\%B]$.chunk_number $= x$ then
        ProcessDupplicateChunk
    else
        ProcessNewChunk
function $P_k$.ProcessDuplicateChunk
    [prune $x$] $\rightarrow$ sender

function $P_k$.ProcessNewChunk
    buffer$[x\%B] = [x,$ origin, chunk$]$
    if sender $\neq S$ then
        debt[sender] = debt[sender] $- 1$
        forward$[P_k]$ = forward$[P_k] \cup$ sender
    for all $P_i \in$ forward[origin] do
        pending$[P_i]$ = pending$[P_i] \cup x$
    for all chunk$_n$umber $\in$ pending[neighbor] do
        buffer$[x\%B] \rightarrow$ neighbor
        pending[neighbor] = pending[neighbor] $\setminus x$
        debt[neighbor] = debt[neighbor] $+ 1$
        if debt[neighbor] $> D$ then
            for all peers_list $\in$ forward do
                peers_list = peers_list $\setminus$ neighbor
    neighbor =

function $P_k$.ProcessControlMsg
    if $x =$ request then
        origin = buffer[chunk$\%B$].origin
        forward[origin] = forward[origin] $\cup$ sender
    if $x =$ prune then
        origin = buffer[chunk$\%B$].origin
        forward[origin] = forward[origin] $\setminus$ sender
    if $x =$ hello then
        Peer.add_neighbor(sender)
    if $x =$ goodbye then
        for all peers_list $\in$ forward do
```

$$\texttt{peers\_list} = \texttt{peers\_list} \setminus \texttt{sender}$$