上层发送数据（将数据插入buffer，并唤醒从buffer 中读取data的线程）

CUDT::sendmsg

CSndUList::update

CSndUList::insert\_

CSndQueue::worker

CSndUList::pop

CUDT::packData

CSndBuffer::readData

CSndUList::remove\_（尽当TTL被设置时，才会从丢失列表中移除。或者在收到对应的ACK时，也会从丢失列表中移除）

CSndUList::insert\_

CChannel::sendto

void\* CRcvQueue::worker(void\* param)

CUDT::checkTimers()

void\* CRcvQueue::worker(void\* param)

// reading next incoming packet, recvfrom returns -1 is nothing has been received

if (self->m\_pChannel->recvfrom(addr, unit->m\_Packet) < 0)

goto TIMER\_CHECK;

id = unit->m\_Packet.m\_iID;

// ID 0 is for connection request, which should be passed to the listening socket or rendezvous sockets

if (0 == id)

else if (id > 0)

if (NULL != (u = self->m\_pHash->lookup(id)))

if (CIPAddress::ipcmp(addr, u->m\_pPeerAddr, u->m\_iIPversion))

if (u->m\_bConnected && !u->m\_bBroken && !u->m\_bClosing)

if (0 == unit->m\_Packet.getFlag()) // 数据包类型

u->processData(unit);

else // 控制包类型

u->processCtrl(unit->m\_Packet);

u->checkTimers();

self->m\_pRcvUList->update(u);

void CUDT::checkTimers()

uint64\_t next\_exp\_time;

if (m\_pCC->m\_bUserDefinedRTO) // 如果定义了超时时间，则使用超时时间

next\_exp\_time = m\_ullLastRspTime + m\_pCC->m\_iRTO \* m\_ullCPUFrequency;

else

{

uint64\_t exp\_int = (m\_iEXPCount \* (m\_iRTT + 4 \* m\_iRTTVar) + m\_iSYNInterval) \* m\_ullCPUFrequency;

if (exp\_int < m\_iEXPCount \* m\_ullMinExpInt)

exp\_int = m\_iEXPCount \* m\_ullMinExpInt;

next\_exp\_time = m\_ullLastRspTime + exp\_int;

}

// 当前时间大于程序设定的下一次检测时间，开始检测

if (currtime > next\_exp\_time)

// sender: Insert all the packets sent after last received acknowledgement into the sender loss list.

// recver: Send out a keep-alive packet

if (m\_pSndBuffer->getCurrBufSize() > 0) // 在特定时间间隔检测到buffer内部有数据，表示数据未及时发送出去。处理：将其加入到丢失列表，以提高其优先级，并且立即启动发送（发送模块会优先发送丢失列表中的数据）

{

if ((CSeqNo::incseq(m\_iSndCurrSeqNo) != m\_iSndLastAck) && (m\_pSndLossList->getLossLength() == 0))

{

// resend all unacknowledged packets on timeout, but only if there is no packet in the loss list

int32\_t csn = m\_iSndCurrSeqNo;

int num = m\_pSndLossList->insert(m\_iSndLastAck, csn);

m\_iTraceSndLoss += num;

m\_iSndLossTotal += num;

}

m\_pCC->onTimeout();

CCUpdate();

// immediately restart transmission立即重启重传

m\_pSndQueue->m\_pSndUList->update(this);

}

else

{

if(keepalive\_cout++ >30){ // 若当前buffer为空，则在检测次数到达一定值时，发送保活包

keepalive\_cout = 0;

sendCtrl(1);

}

}