```
Unit inherited;
procedure tthread_Bittorrent.transferDeal;
var
i:integer;
tran:TBittorrentTransfer;
begin
 for i:=0 to BitTorrentTransfers.count-1 do begin
  tran:=BitTorrentTransfers[i];
  if tran.fstate=dlAllocating then continue;
  transferDeal(tran);
  if terminated then break;
  if (i mod 3)=0 then sleep(1);
 end;
end;
function              TThreadTransfer.transferDeal(transfer:TBittorrentTransfer;
source:tBitTorrentSource):boolean;
var
 er,len_recv,to_recv,previousLen:integer;
 wanted_payload_len:cardinal;
 buffhead:array[0..3] of byte;
begin
result:=false;
//Form1.Memo2.Lines.Add('transferDeal');
try
//reveive and flush
if source.outbuffer.count>0 then begin
 SourceFlush(transfer,source);
 if source.status<>btSourceConnected then exit;
 if source.outbuffer.count>=25 then exit; //try to flush buffer before catching more
requests
end;
if not TCPSocket_CanRead(source.socket.socket,0,er) then begin
 if ((er<>0) and (er<>WSAEWOULDBLOCK)) then begin
   log('Disconnecting source '+source.ipS+' hangup on receive1');
  calcSourceUptime(source);
  SourceDisconnect(source);
 end;
 exit;
end;
// receive 4 byte header
if source.bytes_in_header<4 then begin

len_recv:=TCPSocket_RecvBuffer(source.socket.socket,@source.header[source.bytes
_in_header],4-source.bytes_in_header,er);
 if er=WSAEWOULDBLOCK then exit;
 if er<>0 then begin
   log('Disconnecting source '+source.ipS+' hangup on receive2');
   calcSourceUptime(source);
   SourceDisconnect(source);
   exit;
```

```
 end;
 inc(source.bytes_in_header,len_recv);
 if source.bytes_in_header<4 then exit;
end;
  buffhead[3]:=source.header[0];
  buffhead[2]:=source.header[1];
  buffhead[1]:=source.header[2];
  buffhead[0]:=source.header[3];
  move(buffhead,wanted_payload_len,4);
  {
  wanted_payload_len:=ord(source.header[0]);
  wanted_payload_len:=wanted_payload_len shl 8;
  wanted_payload_len:=wanted_payload_len + ord(source.header[1]);
  wanted_payload_len:=wanted_payload_len shl 8;
  wanted_payload_len:=wanted_payload_len + ord(source.header[2]);
  wanted_payload_len:=wanted_payload_len shl 8;
  wanted_payload_len:=wanted_payload_len + ord(source.header[3]);
 }
  if wanted_payload_len=0 then begin
   source.bytes_in_header:=0; //next packet
   source.inBuffer:='';
   source.lastKeepAliveIn:=tickTransfer;
   exit;
  end;     if wanted_payload_len>BITTORRENT_PIECE_LENGTH+50{9} then begin
      log('Disconnecting source '+source.ipS+' packet receive too big');
      source.status:=btSourceShouldDisconnect;
      exit;
     end;while (cardinal(length(source.inBuffer))<wanted_payload_len) do begin
  to_recv:=wanted_payload_len-cardinal(length(source.inbuffer));
  if to_recv>4096 then to_recv:=4096;

len_recv:=TCPSocket_RecvBuffer(source.socket.socket,@bufferRecvBittorrent,to_re
cv,er);
  if er=WSAEWOULDBLOCK then exit;
  if er<>0 then begin
   log('Disconnecting source '+source.ipS+' hangup on receive3');
   calcSourceUptime(source);
   SourceDisconnect(source);
   exit;
  end;
  if len_recv=0 then begin
   exit;
  end;
  previousLen:=length(source.inBuffer);
  SetLength(source.inBuffer,previousLen+len_recv);
  move(bufferRecvBittorrent,source.inBuffer[previousLen+1],len_recv);

Source_Increase_ReceiveStats(transfer,source,previousLen,len_recv,tickTransfer)
;
  if terminated then exit;
```

```
end;
// if cardinal(length(source.inBuffer))>wanted_payload_len then begin
// end;
 SourceParsePacket(transfer,source);
 source.bytes_in_header:=0;
 source.inBuffer:='';

 if source.status<>btSourceConnected then begin
  result:=false;
  calcSourceUptime(source);

 end else result:=true;
except
end;
end;
function           tthread_bitTorrent.transferDeal(transfer:TBittorrentTransfer;
source:tBitTorrentSource):boolean;
var
 er,len_recv,to_recv,previousLen:integer;
 wanted_payload_len:cardinal;
 buffhead:array[0..3] of byte;
begin
result:=false;
//Form1.Memo2.Lines.Add('transferDeal');
try
//reveive and flush
if source.outbuffer.count>0 then begin
 SourceFlush(transfer,source);
 if source.status<>btSourceConnected then exit;
 if source.outbuffer.count>=25 then exit; //try to flush buffer before catching more
requests
end;
if not TCPSocket_CanRead(source.socket.socket,0,er) then begin
 if ((er<>0) and (er<>WSAEWOULDBLOCK)) then begin
   log('Disconnecting source '+source.ipS+' hangup on receive1');
  calcSourceUptime(source);
  SourceDisconnect(source);
 end;
 exit;
end;
// receive 4 byte header
if source.bytes_in_header<4 then begin

len_recv:=TCPSocket_RecvBuffer(source.socket.socket,@source.header[source.bytes
_in_header],4-source.bytes_in_header,er);
 if er=WSAEWOULDBLOCK then exit;
 if er<>0 then begin
   log('Disconnecting source '+source.ipS+' hangup on receive2');
   calcSourceUptime(source);
   SourceDisconnect(source);
```

```
   exit;
 end;
 inc(source.bytes_in_header,len_recv);
 if source.bytes_in_header<4 then exit;
end;
  buffhead[3]:=source.header[0];
  buffhead[2]:=source.header[1];
  buffhead[1]:=source.header[2];
  buffhead[0]:=source.header[3];
  move(buffhead,wanted_payload_len,4);
  {
  wanted_payload_len:=ord(source.header[0]);
  wanted_payload_len:=wanted_payload_len shl 8;
  wanted_payload_len:=wanted_payload_len + ord(source.header[1]);
  wanted_payload_len:=wanted_payload_len shl 8;
  wanted_payload_len:=wanted_payload_len + ord(source.header[2]);
  wanted_payload_len:=wanted_payload_len shl 8;
  wanted_payload_len:=wanted_payload_len + ord(source.header[3]);
 }
  if wanted_payload_len=0 then begin
   source.bytes_in_header:=0; //next packet
   source.inBuffer:='';
   source.lastKeepAliveIn:=tick;
   exit;
  end;     if wanted_payload_len>BITTORRENT_PIECE_LENGTH+50{9} then begin
      log('Disconnecting source '+source.ipS+' packet receive too big');
      source.status:=btSourceShouldDisconnect;
      exit;
     end;while (cardinal(length(source.inBuffer))<wanted_payload_len) do begin
  to_recv:=wanted_payload_len-cardinal(length(source.inbuffer));
  if to_recv>4096 then to_recv:=4096;

len_recv:=TCPSocket_RecvBuffer(source.socket.socket,@bufferRecvBittorrent,to_re
cv,er);
  if er=WSAEWOULDBLOCK then exit;
  if er<>0 then begin
   log('Disconnecting source '+source.ipS+' hangup on receive3');
   calcSourceUptime(source);
   SourceDisconnect(source);
   exit;
  end;
  if len_recv=0 then begin
   exit;
  end;
  previousLen:=length(source.inBuffer);
  SetLength(source.inBuffer,previousLen+len_recv);
  move(bufferRecvBittorrent,source.inBuffer[previousLen+1],len_recv);
  Source_Increase_ReceiveStats(transfer,source,previousLen,len_recv,tick);
  if terminated then exit;
end;
```

```
// if cardinal(length(source.inBuffer))>wanted_payload_len then begin
// end;
 SourceParsePacket(transfer,source);
 source.bytes_in_header:=0;
 source.inBuffer:='';

 if source.status<>btSourceConnected then begin
  result:=false;
  calcSourceUptime(source);

 end else result:=true;
except
end;
end;
procedure            Source_Increase_ReceiveStats(transfer:TBittorrentTransfer;
Source:TBittorrentSource; previousLen,len_recv:integer; tick:cardinal);
begin
  // increase receive count if it's a piece packet  if length(source.InBuffer)=0
then exit;
   if source.InBuffer[1]=chr(CMD_BITTORRENT_PIECE) then begin
     if previousLen=0 then begin
       if len_recv>9 then inc(source.recv,Len_recv-9);
     end else inc(source.recv,Len_recv);
       if transfer.fstate<>dlPaused then
        if transfer.Fstate<>dlSeeding then transfer.fstate:=dlDownloading;
        source.Snubbed:=false;
        source.lastDataIn:=tick; //good guy
   end;
end;procedure parse_ut_pex(transfer:TBittorrentTransfer; cont:string);
var
 ipC:cardinal;
 portW:word;
 added:integer;
begin
if transfer.fsources.count>=BITTORRENT_MAX_ALLOWED_SOURCES then exit;
 added:=0;
 while (length(cont)>=6) do begin
  ipC:=chars_2_dword(copy(cont,1,4));
  portW:=chars_2_wordRev(copy(cont,5,2));
  delete(cont,1,6);
  transfer.addSource(ipC,portW,'','PEX',false);
  if transfer.fsources.count>=BITTORRENT_MAX_ALLOWED_SOURCES then break;
  inc(added);
  if added>=50 then break;
 end;
end;
procedure
TThreadTransfer.SourceParsePacket(transfer:TBittorrentTransfer;source:TBittorre
ntSource);
var
```

```
 cmdId:byte;
 ind:integer;
begin
try
if transfer.fstate=dlPaused then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;  cmdId:=ord(source.inBuffer[1]);
  delete(source.inBuffer,1,1);case cmdId of
 CMD_BITTORRENT_CHOKE:begin
                       source.weAreChoked:=true;
                       if transfer.isCompleted then exit;

                       RemoveOutGoingRequests(transfer,source);//remove     all
pending 'inUse' requests
                       if not source.isSeeder then
                        if source.SlotType<>ST_OPTIMISTIC then
                         if source.speed_recv>0 then begin
                          source.snubbed:=true;
                            if not source.isChoked then begin
                             source.isChoked:=true;
                             ind:=transfer.UploadSlots.indexof(source);
                             if ind<>-1 then transfer.UploadSlots.delete(ind);

source_AddOutPacket(source,'',CMD_BITTORRENT_CHOKE);
                            end;
                          end;
                      end;
 CMD_BITTORRENT_UNCHOKE:begin
                        if not source.weAreChoked then begin
                         exit;
                        end;
                        RemoveOutGoingRequests(transfer,source);//remove    all
pending 'inUse' requests
                        source.weAreChoked:=false;
                        if   transfer.fstate=dlBittorrentMagnetDiscovery   then
exit;
                        if source.weAreInterested then
                         begin
                           while
(source.outRequests<GetoptimumNumOutRequests(source.speed_recv)) do
                                  if not AskChunk(Transfer,source,tickTransfer)
then break;
                           end;
                        end;
 CMD_BITTORRENT_INTERESTED:begin
                           source.isInterested:=true;
                          //       if       not       source.ischocked       then
ChokeWorstDownload(transfer,source); // a good uploader is now interested, let's
choke our worst downloader(worst downloading uploader)
```

```
                            end;
 CMD_BITTORRENT_NOTINTERESTED:begin
                              source.isInterested:=false;
                              if not source.isSeeder then
                               if not source.isChoked then begin
                                source.isChoked:=true;
                                ind:=transfer.uploadSlots.indexof(source);
                                if ind<>-1 then transfer.uploadSlots.delete(ind);

source_AddOutPacket(source,'',CMD_BITTORRENT_CHOKE);
                               end;
                              end;
 CMD_BITTORRENT_HAVE:UpdateBitField(transfer,source);
 CMD_BITTORRENT_BITFIELD:ResetBitField(transfer,source);
 CMD_BITTORRENT_REQUEST:HandleIncomingRequest(transfer,source);
 CMD_BITTORRENT_PIECE:handleIncomingPiece(transfer,source);
 CMD_BITTORRENT_CANCEL:HandleCancelMessage(transfer,source);
 CMD_BITTORRENT_DHTUDPPORT:mdht_handle_udpport(source);//  dht udp port
 CMD_BITTORRENT_SUGGESTPIECE:Handle_FastPeer_SuggestPiece(transfer,source);
 CMD_BITTORRENT_HAVEALL:Handle_FastPeer_HaveAll(transfer,source);
 CMD_BITTORRENT_HAVENONE:Handle_FastPeer_HaveNone(transfer,source);
 CMD_BITTORRENT_REJECTREQUEST:Handle_FastPeer_RejectRequest(transfer,source);
 CMD_BITTORRENT_ALLOWEDFAST:handle_fastpeer_allowedfast(transfer,source);
 CMD_BITTORRENT_EXTENSION:Handle_ExtensionProtocol_Message(transfer,source);
 end;
except
end;
end;
procedure
tthread_bitTorrent.SourceParsePacket(transfer:TBittorrentTransfer;source:TBitto
rrentSource);
var
 cmdId:byte;
 ind:integer;
begin
try
if transfer.fstate=dlPaused then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;  cmdId:=ord(source.inBuffer[1]);
  delete(source.inBuffer,1,1);case cmdId of
 CMD_BITTORRENT_CHOKE:begin
                       source.weAreChoked:=true;
                       if transfer.isCompleted then exit;

                       RemoveOutGoingRequests(transfer,source);//remove     all
pending 'inUse' requests
                       if not source.isSeeder then
                        if source.SlotType<>ST_OPTIMISTIC then
                         if source.speed_recv>0 then begin
```

```
                            source.snubbed:=true;
                              if not source.isChoked then begin
                               source.isChoked:=true;
                               ind:=transfer.UploadSlots.indexof(source);
                               if ind<>-1 then transfer.UploadSlots.delete(ind);

source_AddOutPacket(source,'',CMD_BITTORRENT_CHOKE);
                              end;
                            end;
                        end;
 CMD_BITTORRENT_UNCHOKE:begin
                          if not source.weAreChoked then begin
                           exit;
                          end;
                          RemoveOutGoingRequests(transfer,source);//remove    all
pending 'inUse' requests
                          source.weAreChoked:=false;
                          if   transfer.fstate=dlBittorrentMagnetDiscovery   then
exit;
                          if source.weAreInterested then
                           begin
                             while
(source.outRequests<GetoptimumNumOutRequests(source.speed_recv)) do
                                    if  not  AskChunk(Transfer,source,tick)  then
break;
                             end;
                          end;
 CMD_BITTORRENT_INTERESTED:begin
                             source.isInterested:=true;
                            //      if      not      source.ischocked      then
ChokeWorstDownload(transfer,source); // a good uploader is now interested, let's
choke our worst downloader(worst downloading uploader)
                             end;
 CMD_BITTORRENT_NOTINTERESTED:begin
                               source.isInterested:=false;
                               if not source.isSeeder then
                                if not source.isChoked then begin
                                 source.isChoked:=true;
                                 ind:=transfer.uploadSlots.indexof(source);
                                 if ind<>-1 then transfer.uploadSlots.delete(ind);

source_AddOutPacket(source,'',CMD_BITTORRENT_CHOKE);
                                end;
                               end;
 CMD_BITTORRENT_HAVE:UpdateBitField(transfer,source);
 CMD_BITTORRENT_BITFIELD:ResetBitField(transfer,source);
 CMD_BITTORRENT_REQUEST:HandleIncomingRequest(transfer,source);
 CMD_BITTORRENT_PIECE:handleIncomingPiece(transfer,source);
 CMD_BITTORRENT_CANCEL:HandleCancelMessage(transfer,source);
 CMD_BITTORRENT_DHTUDPPORT:mdht_handle_udpport(source);//  dht udp port
```

```
 CMD_BITTORRENT_SUGGESTPIECE:Handle_FastPeer_SuggestPiece(transfer,source);
 CMD_BITTORRENT_HAVEALL:Handle_FastPeer_HaveAll(transfer,source);
 CMD_BITTORRENT_HAVENONE:Handle_FastPeer_HaveNone(transfer,source);
 CMD_BITTORRENT_REJECTREQUEST:Handle_FastPeer_RejectRequest(transfer,source);
 CMD_BITTORRENT_ALLOWEDFAST:handle_fastpeer_allowedfast(transfer,source);
 CMD_BITTORRENT_EXTENSION:Handle_ExtensionProtocol_Message(transfer,source);
 end;
except
end;
end;
function Tnt_CreateDirectoryW(lpPathName: PWideChar;
  lpSecurityAttributes: PSecurityAttributes): BOOL;
var Win32PlatformIsUnicode : boolean;
begin
  Win32PlatformIsUnicode := (Win32Platform = VER_PLATFORM_WIN32_NT);
  if Win32PlatformIsUnicode then
    Result    :=    CreateDirectoryW{TNT-ALLOW    CreateDirectoryW}(lpPathName,
lpSecurityAttributes)
  else
    Result                      :=                      CreateDirectoryA{TNT-ALLOW
CreateDirectoryA}(PAnsiChar(AnsiString(lpPathName)), lpSecurityAttributes);
end;
procedure
TThreadTransfer.Handle_ExtensionProtocol_Message(transfer:TBittorrentTransfer;
source:TBittorrentSource);
var
 opCode:byte;
 lenTag:byte;
 lencont:integer;
 tag:string;
 cont:string;
 reqpieceid:integer;
 SavedId:string;
 SavedProgressive:Boolean;
begin
if not source.SupportsExtensions then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;
if length(source.inBuffer)<2 then exit;
 opcode:=ord(source.inBuffer[1]);
 if opcode=OUR_UT_PEX_OPCODE then begin
   delete(source.inBuffer,1,2);
   while (length(source.inBuffer)>10) do begin
    lentag:=strtointdef(copy(source.inBuffer,1,pos(':',source.inBuffer)-1),0);
    if lentag<>5 then break;
    tag:=copy(source.inBuffer,length(inttostr(lentag))+2,lentag);
    if length(tag)<>lentag then begin
     break;
    end;
```

```
    if tag<>'added' then begin
     break;
    end;
     delete(source.inBuffer,1,lentag+length(inttostr(lentag))+1);
    lencont:=strtointdef(copy(source.inBuffer,1,pos(':',source.inBuffer)-1),0);
    if lencont<6 then begin
     break;
    end;
    cont:=copy(source.inBuffer,length(inttostr(lencont))+2,lencont);
    if length(cont)<>lencont then begin
     break;
     end;
     delete(source.inBuffer,1,lencont+length(inttostr(lencont))+1);
    parse_ut_pex(transfer,cont);
    break;
   end;
  exit;
 end;
 if opcode=OUR_UT_METADATA_OPCODE then begin
   delete(source.inBuffer,1,2);
   if pos('8:msg_typeile',source.inBuffer)=0 then begin
    exit;
   end;

cont:=copy(source.inBuffer,pos('5:piecei',source.inBuffer)+8,length(source.inBu
ffer));
   delete(cont,pos('e',cont),length(cont));
   reqpieceid:=strtointdef(cont,-1);
   if reqpieceid=-1 then begin
    exit;
   end;
   if reqpieceid*16384>transfer.ut_metadatasize then begin
    exit;
   end;
   delete(source.inBuffer,1,pos('ee',source.inBuffer)+1);
   if transfer.tempmetastream=nil then exit;
   transfer.tempmetastream.Seek(reqpieceid*16384,soFromBeginning);
   transfer.tempmetastream.Write(source.inBuffer[1],length(source.inBuffer));
   if transfer.tempmetastream.size>=transfer.ut_metadatasize then begin
    //ShowMessage('1: '+transfer.fid);
    SavedId:=transfer.fid;
    SavedProgressive:=transfer.fprogressive;
    transfer.initFrom_ut_Meta;
    transfer.fid:=SavedId;
    transfer.fprogressive:=SavedProgressive;
    GlobTransfer:=transfer;
//     synchronize(update_transfer_visual);
   end else begin
     reqpieceid:=(transfer.tempmetastream.size div 16384);
     source_AddOutPacket(source,
```

```
chr(source.ut_metadata_opcode)+'d8:msg_typei0e5:piecei'+inttostr(reqpieceid)+'e
e',
                          CMD_BITTORRENT_EXTENSION);
   end;
   exit;
 end;
 if opcode=OPCODE_EXTENDED_HANDSHAKE then begin
   if source.port=0 then begin
    if pos('1:pi',source.inBuffer)<>0 then begin
     cont:=copy(source.inBuffer,pos('1:pi',source.inBuffer)+4,6);
     delete(cont,pos('e',cont),length(cont));
      source.port:=strtointdef(cont,0);
    end;
   end;
   cont:=copy(source.inBuffer,pos('6:ut_pexi',source.inBuffer)+9,1);
   source.ut_pex_opcode:=strtointdef(cont,0);
   cont:=copy(source.inBuffer,pos('11:ut_metadatai',source.inBuffer)+15,1);
   source.ut_metadata_opcode:=strtointdef(cont,0);

cont:=copy(source.inBuffer,pos('13:metadata_sizei',source.inBuffer)+17,length(s
ource.inBuffer));
   delete(cont,pos('e',cont),length(cont));
   if                      transfer.ut_metadatasize=0                      then
transfer.ut_metadatasize:=strtointdef(cont,0);
   if (transfer.fstate=dlBittorrentMagnetDiscovery) and
      (transfer.ut_metadatasize>0) then begin

transfer.metafilenameS:=widestrtoutf8str(vars_global.data_Path+'\Data\TempDl\ME
TA_'+bytestr_to_hexstr(transfer.fHashValue)+'.dat');
     tnt_createdirectoryW(pwidechar(vars_global.data_Path+'\Data'),nil);
     tnt_createdirectoryW(pwidechar(vars_global.data_Path+'\Data\TempDl'),nil);
      if                     transfer.tempmetastream=nil                     then
transfer.tempmetastream:=MyFileOpen(utf8strtowidestr(transfer.metafilenameS),AR
ES_OVERWRITE_EXISTING);
      if transfer.tempmetastream=nil then exit;
      reqpieceid:=(transfer.tempmetastream.size div 16384);
      source_AddOutPacket(source,

chr(source.ut_metadata_opcode)+'d8:msg_typei0e5:piecei'+inttostr(reqpieceid)+'e
e',
                          CMD_BITTORRENT_EXTENSION);
   end;
 end;end;
procedure
tthread_bittorrent.Handle_ExtensionProtocol_Message(transfer:TBittorrentTransfe
r; source:TBittorrentSource);
var
 opCode:byte;
 lenTag:byte;
```

```
 lencont:integer;
 tag:string;
 cont:string;
 reqpieceid:integer;
 SavedId:string;
 SavedProgressive:Boolean;
begin
if not source.SupportsExtensions then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;
if length(source.inBuffer)<2 then exit;
 opcode:=ord(source.inBuffer[1]);
 if opcode=OUR_UT_PEX_OPCODE then begin
   delete(source.inBuffer,1,2);
   while (length(source.inBuffer)>10) do begin
    lentag:=strtointdef(copy(source.inBuffer,1,pos(':',source.inBuffer)-1),0);
    if lentag<>5 then break;
    tag:=copy(source.inBuffer,length(inttostr(lentag))+2,lentag);
    if length(tag)<>lentag then begin
     break;
    end;
    if tag<>'added' then begin
     break;
    end;
     delete(source.inBuffer,1,lentag+length(inttostr(lentag))+1);
    lencont:=strtointdef(copy(source.inBuffer,1,pos(':',source.inBuffer)-1),0);
    if lencont<6 then begin
     break;
    end;
    cont:=copy(source.inBuffer,length(inttostr(lencont))+2,lencont);
    if length(cont)<>lencont then begin
     break;
     end;
     delete(source.inBuffer,1,lencont+length(inttostr(lencont))+1);
    parse_ut_pex(transfer,cont);
    break;
   end;
  exit;
 end;
 if opcode=OUR_UT_METADATA_OPCODE then begin
   delete(source.inBuffer,1,2);
   if pos('8:msg_typeile',source.inBuffer)=0 then begin
    exit;
   end;

cont:=copy(source.inBuffer,pos('5:piecei',source.inBuffer)+8,length(source.inBu
ffer));
   delete(cont,pos('e',cont),length(cont));
   reqpieceid:=strtointdef(cont,-1);
```

```
   if reqpieceid=-1 then begin
    exit;
   end;
   if reqpieceid*16384>transfer.ut_metadatasize then begin
    exit;
   end;
   delete(source.inBuffer,1,pos('ee',source.inBuffer)+1);
   if transfer.tempmetastream=nil then exit;
   transfer.tempmetastream.Seek(reqpieceid*16384,soFromBeginning);
   transfer.tempmetastream.Write(source.inBuffer[1],length(source.inBuffer));
   if transfer.tempmetastream.size>=transfer.ut_metadatasize then begin
    //ShowMessage('1: '+transfer.fid);
    SavedId:=transfer.fid;
    SavedProgressive:=transfer.fprogressive;
    transfer.initFrom_ut_Meta;
    transfer.fid:=SavedId;
    transfer.fprogressive:=SavedProgressive;
    GlobTransfer:=transfer;
//     synchronize(update_transfer_visual);
   end else begin
      reqpieceid:=(transfer.tempmetastream.size div 16384);
      source_AddOutPacket(source,

chr(source.ut_metadata_opcode)+'d8:msg_typei0e5:piecei'+inttostr(reqpieceid)+'e
e',
                          CMD_BITTORRENT_EXTENSION);
   end;
   exit;
 end;
 if opcode=OPCODE_EXTENDED_HANDSHAKE then begin
   if source.port=0 then begin
    if pos('1:pi',source.inBuffer)<>0 then begin
     cont:=copy(source.inBuffer,pos('1:pi',source.inBuffer)+4,6);
     delete(cont,pos('e',cont),length(cont));
      source.port:=strtointdef(cont,0);
    end;
   end;
   cont:=copy(source.inBuffer,pos('6:ut_pexi',source.inBuffer)+9,1);
   source.ut_pex_opcode:=strtointdef(cont,0);
   cont:=copy(source.inBuffer,pos('11:ut_metadatai',source.inBuffer)+15,1);
   source.ut_metadata_opcode:=strtointdef(cont,0);

cont:=copy(source.inBuffer,pos('13:metadata_sizei',source.inBuffer)+17,length(s
ource.inBuffer));
   delete(cont,pos('e',cont),length(cont));
   if                    transfer.ut_metadatasize=0                    then
transfer.ut_metadatasize:=strtointdef(cont,0);
   if (transfer.fstate=dlBittorrentMagnetDiscovery) and
      (transfer.ut_metadatasize>0) then begin
```

```
transfer.metafilenameS:=widestrtoutf8str(vars_global.data_Path+'\Data\TempDl\ME
TA_'+bytestr_to_hexstr(transfer.fHashValue)+'.dat');
     tnt_createdirectoryW(pwidechar(vars_global.data_Path+'\Data'),nil);
     tnt_createdirectoryW(pwidechar(vars_global.data_Path+'\Data\TempDl'),nil);
      if                    transfer.tempmetastream=nil                    then
transfer.tempmetastream:=MyFileOpen(utf8strtowidestr(transfer.metafilenameS),AR
ES_OVERWRITE_EXISTING);
      if transfer.tempmetastream=nil then exit;
      reqpieceid:=(transfer.tempmetastream.size div 16384);
      source_AddOutPacket(source,

chr(source.ut_metadata_opcode)+'d8:msg_typei0e5:piecei'+inttostr(reqpieceid)+'e
e',
                          CMD_BITTORRENT_EXTENSION);
   end;
 end;end;
procedure
TThreadTransfer.Handle_FastPeer_SuggestPiece(transfer:TBittorrentTransfer;
source:TBittorrentSource);
begin
exit;
 // Suggest Piece: <len=0x0005><op=0x0D><index>
if not source.SupportsFastPeer then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;
end;
procedure
tthread_bitTorrent.Handle_FastPeer_SuggestPiece(transfer:TBittorrentTransfer;
source:TBittorrentSource);
begin
exit;
 // Suggest Piece: <len=0x0005><op=0x0D><index>
if not source.SupportsFastPeer then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;
end;
procedure
TThreadTransfer.handle_fastpeer_allowedfast(transfer:TBittorrentTransfer;
source:TBittorrentSource);
begin
exit;
 // Suggest Piece: <len=0x0005><op=0x0D><index>
if not source.SupportsFastPeer then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;
end;
procedure
```

```
tthread_bitTorrent.handle_fastpeer_allowedfast(transfer:TBittorrentTransfer;
source:TBittorrentSource);
begin
exit;
 // Suggest Piece: <len=0x0005><op=0x0D><index>
if not source.SupportsFastPeer then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;
end;
procedure TThreadTransfer.Handle_FastPeer_HaveAll(transfer:TBittorrentTransfer;
source:TBittorrentSource);
var
i:integer;
piece:TBitTorrentChunk;
begin
exit;
// Have All: <len=0x0001><op=0x0E>
if not source.SupportsFastPeer then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;if                          source.bitfield=nil                          then
source.bitfield:=tbittorrentBitfield.create(length(transfer.fpieces));
for i:=0 to high(source.bitfield.bits) do source.bitfield.bits[i]:=true;for i:=0
to high(transfer.Fpieces) do begin
 piece:=transfer.FPieces[i];
 if piece.checked then continue;
      if not source.weAreInterested then begin // we are interested, let remote peer
know
        source_AddOutPacket(source,'',CMD_BITTORRENT_INTERESTED);
        source.weAreInterested:=true;
      end;
        break;
end;
 source.progress:=100;
 CalcChunksPopularity(transfer);
 source.changedVisualBitField:=true;
 transfer.CalculateLeechsSeeds;
if transfer.isCompleted then
 if source.isSeeder then source.status:=btSourceShouldRemove;
end;
procedure
tthread_bitTorrent.Handle_FastPeer_HaveAll(transfer:TBittorrentTransfer;
source:TBittorrentSource);
var
i:integer;
piece:TBitTorrentChunk;
begin
exit;
// Have All: <len=0x0001><op=0x0E>
```

```
if not source.SupportsFastPeer then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;if                          source.bitfield=nil                          then
source.bitfield:=tbittorrentBitfield.create(length(transfer.fpieces));
for i:=0 to high(source.bitfield.bits) do source.bitfield.bits[i]:=true;for i:=0
to high(transfer.Fpieces) do begin
 piece:=transfer.FPieces[i];
 if piece.checked then continue;
      if not source.weAreInterested then begin // we are interested, let remote peer
know
        source_AddOutPacket(source,'',CMD_BITTORRENT_INTERESTED);
        source.weAreInterested:=true;
      end;
        break;
end;
 source.progress:=100;
 CalcChunksPopularity(transfer);
 source.changedVisualBitField:=true;
 transfer.CalculateLeechsSeeds;
if transfer.isCompleted then
 if source.isSeeder then source.status:=btSourceShouldRemove;
end;
procedure TThreadTransfer.Handle_FastPeer_HaveNone(transfer:TBittorrentTransfer;
source:TBittorrentSource);
var
i:integer;
//piece:TBitTorrentChunk;
begin
exit;
 // Have None: <len=0x0001><op=0x0F>
if not source.SupportsFastPeer then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;
if                            source.bitfield=nil                            then
source.bitfield:=tbittorrentBitfield.create(length(transfer.fpieces));
for i:=0 to high(source.bitfield.bits) do source.bitfield.bits[i]:=false;
 if source.weAreInterested then begin // we are interested, let remote peer know
  source_AddOutPacket(source,'',CMD_BITTORRENT_NOTINTERESTED);
  source.weAreInterested:=false;
 end;
 source.progress:=0;
 CalcChunksPopularity(transfer);
 source.changedVisualBitField:=true;
 transfer.CalculateLeechsSeeds;
end;
procedure
tthread_bitTorrent.Handle_FastPeer_HaveNone(transfer:TBittorrentTransfer;
source:TBittorrentSource);
```

```
var
i:integer;
//piece:TBitTorrentChunk;
begin
exit;
 // Have None: <len=0x0001><op=0x0F>
if not source.SupportsFastPeer then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;
if                              source.bitfield=nil                              then
source.bitfield:=tbittorrentBitfield.create(length(transfer.fpieces));
for i:=0 to high(source.bitfield.bits) do source.bitfield.bits[i]:=false;
 if source.weAreInterested then begin // we are interested, let remote peer know
  source_AddOutPacket(source,'',CMD_BITTORRENT_NOTINTERESTED);
  source.weAreInterested:=false;
 end;
 source.progress:=0;
 CalcChunksPopularity(transfer);
 source.changedVisualBitField:=true;
 transfer.CalculateLeechsSeeds;
end;
procedure
TThreadTransfer.Handle_FastPeer_RejectRequest(transfer:TBittorrentTransfer;
source:TBittorrentSource);
var
 pieceindex,wantedlen,offset:cardinal;
begin
exit;
 // Reject Request: <len=0x000D><op=0x10><index><begin><offset>
if not source.SupportsFastPeer then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end; pieceindex:=chars_2_dwordRev(copy(source.inBuffer,1,4));
 offset:=chars_2_dwordRev(copy(source.inBuffer,5,4));
 wantedlen:=chars_2_dwordRev(copy(source.inBuffer,9,4));

 RemoveoutGointRequest(transfer,
                       source,
                       pieceindex,
                       offset,
                       wantedlen);end;
procedure
tthread_bitTorrent.Handle_FastPeer_RejectRequest(transfer:TBittorrentTransfer;
source:TBittorrentSource);
var
 pieceindex,wantedlen,offset:cardinal;
begin
exit;
 // Reject Request: <len=0x000D><op=0x10><index><begin><offset>
```

```
if not source.SupportsFastPeer then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end; pieceindex:=chars_2_dwordRev(copy(source.inBuffer,1,4));
 offset:=chars_2_dwordRev(copy(source.inBuffer,5,4));
 wantedlen:=chars_2_dwordRev(copy(source.inBuffer,9,4));

 RemoveoutGointRequest(transfer,
                       source,
                       pieceindex,
                       offset,
                       wantedlen);end;
procedure   TThreadTransfer.RemoveoutGointRequest(transfer:tbittorrentTransfer;
source:tbittorrentSource;         pieceindex:cardinal;         offset:cardinal;
wantedlen:cardinal);
var
i:integer;
request:precord_BitTorrentoutgoing_request;
begin
for i:=0 to transfer.outgoingRequests.Count-1 do begin
 request:=transfer.outgoingRequests[i];
 if longint(request^.source)<>longint(source) then continue;
 if cardinal(request^.index)<>pieceindex then continue;
 if request^.offset<>offset then continue;
 //if          request^.wantedlen<>wantedlen          then          continue;
transfer.outgoingRequests.delete(i);
    freeMem(request,sizeof(record_BitTorrentoutgoing_request));
    break;
end;
end;
procedure tthread_bitTorrent.RemoveoutGointRequest(transfer:tbittorrentTransfer;
source:tbittorrentSource;         pieceindex:cardinal;         offset:cardinal;
wantedlen:cardinal);
var
i:integer;
request:precord_BitTorrentoutgoing_request;
begin
for i:=0 to transfer.outgoingRequests.Count-1 do begin
 request:=transfer.outgoingRequests[i];
 if longint(request^.source)<>longint(source) then continue;
 if cardinal(request^.index)<>pieceindex then continue;
 if request^.offset<>offset then continue;
 //if          request^.wantedlen<>wantedlen          then          continue;
transfer.outgoingRequests.delete(i);
    freeMem(request,sizeof(record_BitTorrentoutgoing_request));
    break;
end;
end;
procedure   TThreadTransfer.HandleIncomingRequest(transfer:TBittorrentTransfer;
source:TBittorrentSource);
```

```
var
index,offset,wlen:cardinal;
piece:TBitTorrentChunk;
er:integer;
rem:int64;
buffer:array[0..16383] of char;
str:string;
begin
try
if transfer.fstate=dlPaused then exit;
if length(source.inBuffer)<12 then exit;
index:=chars_2_dwordRev(copy(source.inBuffer,1,4));
offset:=chars_2_dwordRev(copy(source.inBuffer,5,4));
wlen:=chars_2_dwordRev(copy(source.inBuffer,9,4));if
wlen>BITTORRENT_PIECE_LENGTH then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;
if index>cardinal(high(transfer.fpieces)) then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;
piece:=transfer.fpieces[index];
if not piece.checked then begin
 exit;
end;
if source.isChoked then begin
 exit;
end;
 CancelOutGoingPiece(transfer,source,index,offset); //cancel previous outgoing
requests

 transfer.read((int64(index)*int64(transfer.fpiecelength))+int64(offset),
               @buffer,
               wlen,
               rem,
               er);

 if rem<>0 then begin
   source.status:=btSourceShouldDisconnect;
  exit;
 end;
 SetLength(str,wlen-rem);
 move(buffer,str[1],length(str));
 source_AddOutPacket(source,int_2_dword_stringRev(index)+
                            int_2_dword_stringRev(offset)+
                            str,
                            CMD_BITTORRENT_PIECE,
                            false,
                            index,
```

```
                              offset,
                              wlen);except
end;
end;
procedure tthread_bitTorrent.HandleIncomingRequest(transfer:TBittorrentTransfer;
source:TBittorrentSource);
var
index,offset,wlen:cardinal;
piece:TBitTorrentChunk;
er:integer;
rem:int64;
buffer:array[0..16383] of char;
str:string;
begin
try
if transfer.fstate=dlPaused then exit;
if length(source.inBuffer)<12 then exit;
index:=chars_2_dwordRev(copy(source.inBuffer,1,4));
offset:=chars_2_dwordRev(copy(source.inBuffer,5,4));
wlen:=chars_2_dwordRev(copy(source.inBuffer,9,4));if
wlen>BITTORRENT_PIECE_LENGTH then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;
if index>cardinal(high(transfer.fpieces)) then begin
 source.status:=btSourceShouldDisconnect;
 exit;
end;
piece:=transfer.fpieces[index];
if not piece.checked then begin
 exit;
end;
if source.isChoked then begin
 exit;
end;
 CancelOutGoingPiece(transfer,source,index,offset); //cancel previous outgoing
requests

 transfer.read((int64(index)*int64(transfer.fpiecelength))+int64(offset),
               @buffer,
               wlen,
               rem,
               er);

 if rem<>0 then begin
   source.status:=btSourceShouldDisconnect;
  exit;
 end;
 SetLength(str,wlen-rem);
 move(buffer,str[1],length(str));
```

```
 source_AddOutPacket(source,int_2_dword_stringRev(index)+
                             int_2_dword_stringRev(offset)+
                             str,
                             CMD_BITTORRENT_PIECE,
                             false,
                             index,
                             offset,
                             wlen);except
end;
end;
procedure            RemoveOutGoingRequestForPiece(transfer:TBittorrentTransfer;
index:integer);
var
i:integer;
request:precord_BitTorrentoutgoing_request;
begin
i:=0;
while (i<transfer.outgoingRequests.Count) do begin
 request:=transfer.outgoingRequests[i];
 if request^.index<>index then begin
  inc(i);
  continue;
 end;
   Source_AddOutPacket(transfer,
                       request^.source,

int_2_dword_stringRev(request^.index)+int_2_dword_stringRev(request^.offset)+in
t_2_dword_stringRev(request^.wantedLen),
                       CMD_BITTORRENT_CANCEL,
                       true,
                       request^.index,
                       request^.offset,
                       request^.wantedLen);
    transfer.outgoingRequests.delete(i);
    freeMem(request,sizeof(record_BitTorrentoutgoing_request));
end;
end;procedure      CancelOutGoingRequestsForPiece(transfer:TBitTorrentTransfer;
Source:TBittorrentSource; index:cardinal; offset:Cardinal);
var
i:integer;
request:precord_BitTorrentoutgoing_request;
tmpSource:TBitTorrentSource;
begin
i:=0;
while (i<transfer.outgoingRequests.Count) do begin
 request:=transfer.outgoingRequests[i];
 if cardinal(request^.index)<>index then begin
  inc(i);
  continue;
 end;
```

```
 if request^.offset<>offset then begin
   inc(i);
   continue;
 end;
  // if we sent this to another source send a cancel packet for this piece
  if longint(request^.source)<>longint(source) then begin
   tmpSource:=FindSourceFromID(transfer,request^.source);
    if tmpSource<>nil then begin
      if Source_PeekRequest_InIncomingBuffer(tmpsource,request) then begin
      end else begin
       Source_AddOutPacket(tmpSource,

int_2_dword_stringRev(request^.index)+int_2_dword_stringRev(request^.offset)+in
t_2_dword_stringRev(request^.wantedLen),
                           CMD_BITTORRENT_CANCEL,
                           true,
                           request^.index,
                           request^.offset,
                           request^.wantedLen);
      end;
    end;
  end;

    transfer.outgoingRequests.delete(i);
    freeMem(request,sizeof(record_BitTorrentoutgoing_request));
 end;
end;
procedure     TThreadTransfer.handleIncomingPiece(transfer:TBittorrentTransfer;
source:TBittorrentSource);
var
index,
offset:cardinal;
LenData:integer;
piece:TBitTorrentChunk;
rem:int64;
er:integer;
tracker:tbittorrentTracker;
begin
try
if transfer.fstate=dlPaused then exit;
if transfer.fstate=dlSeeding then exit;
if length(source.inBuffer)<9 then begin
 exit;
end;
//log('handleIncomingPiece: 1');
if source.weAreChoked then begin //should we care?
 exit;
end;
//log('handleIncomingPiece: 2');
index:=chars_2_dwordRev(copy(source.inBuffer,1,4));
```

```
offset:=chars_2_dwordRev(copy(source.inBuffer,5,4));
if index>cardinal(high(transfer.fpieces)) then begin
 exit;
end;
 //log('handleIncomingPiece: 3');
 CancelOutGoingRequestsForPiece(transfer,source,index,offset);
LenData:=length(source.inBuffer)-8;
 piece:=transfer.fpieces[index];
 if (offset div BITTORRENT_PIECE_LENGTH)>cardinal(high(piece.pieces)) then begin
  exit;
 end;
 //log('handleIncomingPiece: 4');
 if piece.pieces[offset div BITTORRENT_PIECE_LENGTH] then begin
  exit;
 end;
 //log('handleIncomingPiece: 5');
 if lenData<>BITTORRENT_PIECE_LENGTH then begin
   if piece.findex<>cardinal(high(transfer.fpieces)) then begin
    log('Disconnecting source '+source.ipS+' sent us wrong piecelen');
    source.status:=btSourceShouldRemove;
    exit;
   end;
 end;
 //log('handleIncomingPiece: 6');

transfer.write((int64(piece.findex)*int64(transfer.fpieceLength))+int64(offset)
,
                 @source.inbuffer[9],
                 lenData,
                 rem,
                 er);
 //log('handleIncomingPiece: 7');
  if rem<>0 then begin
   exit;
  end;
 //log('handleIncomingPiece: 8');
 piece.pieces[offset div BITTORRENT_PIECE_LENGTH]:=true;
 inc(piece.fprogress,lenData);
 if source.outRequests>=1 then dec(source.outRequests);
 //log('handleIncomingPiece: 9');
 if             transfer.tempDownloaded+LenData<=transfer.fsize             then
inc(transfer.tempDownloaded,lenData);
 inc(loc_downloadedBytes,lenData);
 //log('handleIncomingPiece: 10');
 if piece.fprogress=piece.fsize then begin // time to check SHA1
   if transfer.hashFails>=NUMMAX_TRANSFER_HASHFAILS then begin
    if piece=source.assignedChunk then begin
     source.assignedChunk:=nil;
     piece.assignedSource:=nil;
    end;
```

```
   end;
   //log('handleIncomingPiece: 11');
   piece.check;
   //log('handleIncomingPiece: 12');
   if piece.checked then begin
    RemoveOutGoingRequestForPiece(transfer,piece.findex);
    if            transfer.hashFails>=NUMMAX_TRANSFER_HASHFAILS            then
inc(source.blocksReceived);
    //log('handleIncomingPiece: 13');
    transfer.changedVisualBitField:=true;
    BroadcastHave(transfer,piece);
    //log('handleIncomingPiece: 14');
    if transfer.isCompleted then begin
    //log('handleIncomingPiece: 15');
     DisconnectSeeders(transfer);
     SetAllNotinterested(transfer);
     transfer.DoComplete;
    //log('handleIncomingPiece: 16');
     if transfer.trackers.count>0 then begin
      tracker:=transfer.trackers[transfer.trackerIndex];
      tracker.next_poll:=0; //send notification to tracker
     end;
     transfer.fstate:=dlSeeding;
      GlobTransfer:=transfer;
//       synchronize(CompleteVisualTransfer);
    end;
    if source.weAreInterested then areWeStillInterested(transfer,source);
   end else begin
    dec(transfer.tempDownloaded,piece.fsize);
    inc(transfer.hashFails);
    if transfer.hashFails>NUMMAX_TRANSFER_HASHFAILS then begin
     inc(source.hashFails);
       if source.hashFails>=NUMMAX_SOURCE_HASHFAILS then begin
         log('Disconnecting source '+source.ipS+' too many hashfails');
        source.status:=btSourceShouldRemove;
        btcore.AddBannedIp(transfer,source.ipC);
        exit;
       end;
    end;
   end;
 end;
  //log('handleIncomingPiece: 17');
  if transfer.fstate=dlBittorrentMagnetDiscovery then exit;
  if source.weAreInterested then begin
   while   (source.outRequests<GetoptimumNumOutRequests(source.speed_recv))   do
begin
    if not AskChunk(Transfer,source,tickTransfer) then break; //ask another piece
   end;
  end;
  //log('handleIncomingPiece: 18');
```

```
 except
 end;
end;
procedure  tthread_bitTorrent.handleIncomingPiece(transfer:TBittorrentTransfer;
source:TBittorrentSource);
var
index,
offset:cardinal;
LenData:integer;
piece:TBitTorrentChunk;
rem:int64;
er:integer;
tracker:tbittorrentTracker;
begin
try
if transfer.fstate=dlPaused then exit;
if transfer.fstate=dlSeeding then exit;
if length(source.inBuffer)<9 then begin
 exit;
end;
//log('handleIncomingPiece: 1');
if source.weAreChoked then begin //should we care?
 exit;
end;
//log('handleIncomingPiece: 2');
index:=chars_2_dwordRev(copy(source.inBuffer,1,4));
offset:=chars_2_dwordRev(copy(source.inBuffer,5,4));
if index>cardinal(high(transfer.fpieces)) then begin
 exit;
end;
 log('handleIncomingPiece: 3');
 CancelOutGoingRequestsForPiece(transfer,source,index,offset);
LenData:=length(source.inBuffer)-8;
 piece:=transfer.fpieces[index];
 if (offset div BITTORRENT_PIECE_LENGTH)>cardinal(high(piece.pieces)) then begin
  exit;
 end;
 log('handleIncomingPiece: 4');
 if piece.pieces[offset div BITTORRENT_PIECE_LENGTH] then begin
  exit;
 end;
 log('handleIncomingPiece: 5');
 if lenData<>BITTORRENT_PIECE_LENGTH then begin
   if piece.findex<>cardinal(high(transfer.fpieces)) then begin
    log('Disconnecting source '+source.ipS+' sent us wrong piecelen');
    source.status:=btSourceShouldRemove;
    exit;
   end;
 end;
 log('handleIncomingPiece: 6');
```

```
transfer.write((int64(piece.findex)*int64(transfer.fpieceLength))+int64(offset)
,
                  @source.inbuffer[9],
                  lenData,
                  rem,
                  er);
 log('handleIncomingPiece: 7');
  if rem<>0 then begin
   exit;
  end;
 //log('handleIncomingPiece: 8');
 piece.pieces[offset div BITTORRENT_PIECE_LENGTH]:=true;
 inc(piece.fprogress,lenData);
 if source.outRequests>=1 then dec(source.outRequests);
 //log('handleIncomingPiece: 9');
 if             transfer.tempDownloaded+LenData<=transfer.fsize              then
inc(transfer.tempDownloaded,lenData);
 inc(loc_downloadedBytes,lenData);
 //log('handleIncomingPiece: 10');
 if piece.fprogress=piece.fsize then begin // time to check SHA1
   if transfer.hashFails>=NUMMAX_TRANSFER_HASHFAILS then begin
    if piece=source.assignedChunk then begin
     source.assignedChunk:=nil;
     piece.assignedSource:=nil;
    end;
   end;
   //log('handleIncomingPiece: 11');
   piece.check;
   //log('handleIncomingPiece: 12');
   if piece.checked then begin
    RemoveOutGoingRequestForPiece(transfer,piece.findex);
    if             transfer.hashFails>=NUMMAX_TRANSFER_HASHFAILS             then
inc(source.blocksReceived);
    //log('handleIncomingPiece: 13');
    transfer.changedVisualBitField:=true;
    BroadcastHave(transfer,piece);
    //log('handleIncomingPiece: 14');
    if transfer.isCompleted then begin
    //log('handleIncomingPiece: 15');
     DisconnectSeeders(transfer);
     SetAllNotinterested(transfer);
     transfer.DoComplete;
    //log('handleIncomingPiece: 16');
     if transfer.trackers.count>0 then begin
      tracker:=transfer.trackers[transfer.trackerIndex];
      tracker.next_poll:=0; //send notification to tracker
     end;
     transfer.fstate:=dlSeeding;
      GlobTransfer:=transfer;
```

```
//      synchronize(CompleteVisualTransfer);
    end;
    if source.weAreInterested then areWeStillInterested(transfer,source);
   end else begin
    dec(transfer.tempDownloaded,piece.fsize);
    inc(transfer.hashFails);
    if transfer.hashFails>NUMMAX_TRANSFER_HASHFAILS then begin
     inc(source.hashFails);
       if source.hashFails>=NUMMAX_SOURCE_HASHFAILS then begin
         log('Disconnecting source '+source.ipS+' too many hashfails');
        source.status:=btSourceShouldRemove;
        btcore.AddBannedIp(transfer,source.ipC);
        exit;
       end;
    end;
   end;
 end;
  //log('handleIncomingPiece: 17');
  if transfer.fstate=dlBittorrentMagnetDiscovery then exit;
  if source.weAreInterested then begin
   while   (source.outRequests<GetoptimumNumOutRequests(source.speed_recv))   do
begin
    if not AskChunk(Transfer,source,tick) then break; //ask another piece
   end;
  end;
  //log('handleIncomingPiece: 18');
 except
 end;
end;
procedure  DisconnectSeeders(transfer:TBitTorrentTransfer);//download  completed
we no longer need seeders
var
i:integer;
source:TBitTorrentSource;
begin
for i:=0 to transfer.fsources.count-1 do begin
  source:=transfer.fsources[i];
  if source.progress<100 then continue;
  source.status:=btSourceShouldRemove;
end;
end;
procedure
TThreadTransfer.SetAllNotinterested(transfer:TBitTorrentTransfer);//download
completed we no longer need seeders
var
i:integer;
source:TBitTorrentSource;
begin
 for i:=0 to transfer.fsources.count-1 do begin
  source:=transfer.fsources[i];
```

```
  if source.status<>btSourceConnected then continue;
   if source.isInterested then begin
     source.isInterested:=false;
     Source_AddOutPacket(source,'',CMD_BITTORRENT_NOTINTERESTED);
   end;
 end;
end;
procedure
tthread_bittorrent.SetAllNotinterested(transfer:TBitTorrentTransfer);//download
completed we no longer need seeders
var
i:integer;
source:TBitTorrentSource;
begin
 for i:=0 to transfer.fsources.count-1 do begin
  source:=transfer.fsources[i];
  if source.status<>btSourceConnected then continue;
   if source.isInterested then begin
     source.isInterested:=false;
     Source_AddOutPacket(source,'',CMD_BITTORRENT_NOTINTERESTED);
   end;
 end;
end;
function         DropWorstConnectedInactiveSource(transfer:TBitTorrentTransfer;
source:TBitTorrentSource; tick:cardinal):boolean;
var
i:integer;
tmpSource:TBitTorrentSource;
begin
result:=false;
if                            transfer.isCompleted                              then
transfer.fsources.sort(BitTorrentSortWorstForaSeederInactiveSourceFirst)
 else
transfer.fsources.sort(BitTorrentSortWorstForaLeecherInactiveSourceFirst);

for i:=0 to transfer.fsources.count-1 do begin
 tmpSource:=transfer.fsources[i];
 if tmpSource=source then continue;
 if tmpSource.status<>btSourceConnected then continue;
 if tick-tmpSource.handShakeTick<5*MINUTE then continue;//minimum threshold
 tmpSource.status:=btSourceShouldDisconnect;
 result:=true;
 break;
end;
end;
procedure BroadcastHave(transfer:TBitTorrentTransfer; piece:TBitTorrentChunk);
var
i:integer;
source:TBittorrentSource;
str:string;
```

```
begin
str:=int_2_dword_stringRev(piece.findex);

for i:=0 to transfer.fsources.count-1 do begin
 source:=transfer.fsources[i];
 if source.status<>btSourceConnected then continue;
 //if source.bitfield<>nil then
  //if source.bitfield.bits[piece.index] then continue; //already have this piece,
don't send my have message?
   source_AddOutPacket(source,str,CMD_BITTORRENT_HAVE);
end;
end;
function  ChoseAnyChunk(transfer:TBitTorrentTransfer;  source:TBitTorrentSource;
var SuggestedFreeOffSetIndex:integer):TBittorrentChunk;
var
i:integer;
piece:TBitTorrentChunk;
begin
result:=nil;
 for i:=0 to high(transfer.fpieces) do begin
  piece:=transfer.fpieces[i];
  if piece.checked then continue;
  if not piece.downloadable then continue; //this chunk is related to a file we do
not want
  if not source.bitfield.bits[i] then continue;
     SuggestedFreeOffSetIndex:=FindAnyPieceMissing(transfer,piece);
     if SuggestedFreeOffSetIndex=-1 then continue;
      result:=piece;
      exit;
 end;
end;
function                      ChoseIncompleteChunk(transfer:TBitTorrentTransfer;
source:TBitTorrentSource;                                                    var
SuggestedFreeOffSetIndex:integer):TBittorrentChunk;
var
i:integer;
piece:TBitTorrentChunk;
begin
result:=nil;
 for i:=0 to high(transfer.fpieces) do begin
  piece:=transfer.fpieces[i];
  if piece.checked then continue;
  if not piece.downloadable then continue; //this chunk is related to a file we do
not want
  if not source.bitfield.bits[i] then continue;
  if piece.fprogress=0 then continue;   if transfer.isEndGameMode then begin

SuggestedFreeOffSetIndex:=FindPieceNotRequestedBySource(transfer,source,piece);
     if SuggestedFreeOffSetIndex=-1 then continue;
   end else begin
```

```
     if piece.assignedSource<>nil then continue;

SuggestedFreeOffSetIndex:=FindPieceNotRequestedByAnySource(transfer,piece);
     if SuggestedFreeOffSetIndex=-1 then continue;
   end;
      result:=piece;
      exit;
 end;
end;
procedure SendPexHandshake(source:tbittorrentSource);
begin
source_AddOutPacket(source,
                          chr(0)+'d1:ei0e1:md'+
                                '6:ut_pexi'+inttostr(OUR_UT_PEX_OPCODE)+'e'+

'11:ut_metadatai'+inttostr(OUR_UT_METADATA_OPCODE)+'e'+

'e1:pi'+inttostr(vars_global.myport)+'e1:v'+inttostr(1+length(AGENT_NAME2)+leng
th(vars_global.versioneares))+':'+AGENT_NAME2+'
'+vars_global.versioneares+'6:yourip4:'+int_2_dword_string(source.ipC)+'e',
                          CMD_BITTORRENT_EXTENSION);
end;
function                        FindAnyPieceMissing(transfer:TBitTorrentTransfer;
piece:TBitTorrentchunk):integer;
var
i:integer;
begin
result:=-1;
 if not piece.pieces[random(length(piece.pieces))] then begin
  result:=i;
  exit;
 end;
 for i:=0 to high(piece.pieces) do begin
   if piece.pieces[i] then continue;
   result:=i;
   exit;
 end;
end;
function         FindPieceNotRequestedByAnySource(transfer:TBitTorrentTransfer;
piece:TBitTorrentchunk):integer;
var
i,h:integer;
cmpOffset:cardinal;
request:precord_BitTorrentoutgoing_request;
found:boolean;
begin
result:=-1;
 for i:=0 to high(piece.pieces) do begin
   if piece.pieces[i] then continue;
   cmpOffset:=i*BITTORRENT_PIECE_LENGTH;
```

```
    found:=false;
    for h:=0 to transfer.outGoingRequests.count-1 do begin
     request:=transfer.outGoingRequests[h];
      if cardinal(request^.index)<>piece.findex then continue;
      if request^.offset<>cmpOffset then continue;
      found:=true;
      break;
    end;
    if not found then begin
     result:=i;
     exit;
    end;
 end;
end;
globSource:=source;
 GlobTransfer:=transfer;
// synchronize(deleteVisualGlobSource);
 if source.bitfield<>nil then CalcChunksPopularity(transfer); // must perform
before source freeing
 source.free;
 CalcNumConnected(transfer);
 transfer.CalculateLeechsSeeds;
end;
procedure TThreadTransfer.execute;
var
i,er,len,hi:integer;
source,tmpSource:tbittorrentSource;
str:string;
buffer:array[0..67] of char;
timeint:cardinal;
PortStr:string;
begin
FreeOnTerminate:=False;
repeat
tickTransfer:=GetTickCount;
try
i:=0;
EnterCriticalSection(TrackerCriticalSection);
try
while (i<transfer.fsources.count) do begin
 if terminated then break;
 source:=transfer.fsources[i];
 case source.status of
    btSourceShouldDisconnect:begin
      DisconnectSource(transfer,source,true);
      inc(i);
      continue;
    end;
    btSourceShouldRemove:begin
      transfer.fsources.delete(i);
```

```
      RemoveSource(transfer,source);
     continue;
    end;

    btSourceConnected:begin
      while transferDeal(transfer,source) do ;
      inc(i);
      continue;
    end;
    btSourceIdle:begin
      if transfer.fstate=dlPaused then begin
       inc(i);
       continue;
      end;
      if  transfer.numConnected>=BITTORENT_MAXNUMBER_CONNECTION_ESTABLISH  then
begin //no need to connect to more sources
       inc(i);
       continue;
      end;
      if GetNumConnecting(transfer)>=MAX_OUTGOING_ATTEMPTS then begin
       inc(i);
       continue;
      end;
      if                          (source.lastAttempt<>0)                          and
(tickTransfer-source.lastAttempt<BTSOURCE_CONN_ATTEMPT_INTERVAL) then begin
       inc(i);
       continue;
      end;
      if transfer.fErrorCode<>0 then begin
//       Form1.Memo2.Lines.Add('transferDealExit');
       Break;//exit;
      end;

      if transfer.isCompleted then
       if source.isSeeder then begin  //this source is a seeder, connect to leechers
only, now that data has been downloaded...
        inc(i);
        continue;
       end;
      source.lastAttempt:=tickTransfer;
      if source.socket<>nil then source.socket.free;
      source.ClearOutBuffer;
      source.inbuffer:='';
      source.socket:=TTCPBlockSocket.create(true);
      source.socket.block(false);
      //helper_sockets.assign_proxy_settings(source.socket);
     if vars_global.socks_type=SocTNone then begin
      source.socket.SocksIP:='';
      source.socket.SocksPort:='0';
     end else begin
```

```
      source.socket.FLastTime:=gettickcount;    //per    vari    timeout    in
TCPSocket_connesso()
      source.socket.SocksIp:=vars_global.socks_ip;
      source.socket.SocksPort:=inttostr(vars_global.socks_port);
      if vars_global.socks_type=SocTSock5 then begin
        source.socket.SocksType:=ST_Socks5;
        source.socket.SocksUsername:=vars_global.socks_username;
        source.socket.SocksPassword:=vars_global.socks_password;
      end else source.socket.SocksType:=ST_Socks4;
      source.socket.FStatoConn:=PROXY_InConnessione;
     end;
      source.tick:=tickTransfer;
      source.status:=btSourceConnecting;
      source.IsIncomingConnection:=false;
      try
        PortStr:=inttostr(source.port);
      except
      end;
      source.socket.connect(source.ipS,PortStr);
      GlobTransfer:=transfer;
      globSource:=source;
//      synchronize(updateVisualGlobsource);
    end;    btSourceConnecting:begin
      if (transfer.fsources.count>=50) or
         (transfer.numConnected>15)        then        timeint:=5000        else
timeint:=TIMEOUTTCPCONNECTION;
      if tickTransfer-source.tick>timeint then begin
        SourceAddFailedAttempt(transfer,source);
        inc(i);
        continue;
      end;
      er:=TCPSocket_ISConnected(source.socket);
      if er=WSAEWOULDBLOCK then begin
       inc(i);
       continue;
      end;
      if ((er<>0) and (er<>WSAEWOULDBLOCK)) then begin
        SourceAddFailedAttempt(transfer,source);
        inc(i);
        continue;
      end;
      str:=STR_BITTORRENT_PROTOCOL_HANDSHAKE+
           STR_BITTORRENT_PROTOCOL_EXTENSIONS+
           transfer.fhashvalue+
           thread_bittorrent.mypeerID;
//      ShowMessage(str);
      TCPSocket_SendBuffer(source.socket.socket,pchar(str),length(str),er);
      if er=WSAEWOULDBLOCK then begin
       inc(i);
       continue;
```

```
      end;
      if er<>0 then begin
       SourceAddFailedAttempt(transfer,source);
       inc(i);
       continue;
      end;
      source.status:=btSourceReceivingHandshake;
      source.tick:=tickTransfer;
      GlobTransfer:=transfer;
      globSource:=source;
//      synchronize(updateVisualGlobsource);
    end;
    btSourceReceivingHandshake:begin
      if (transfer.fsources.count>=50) or
         (transfer.numConnected>15)        then        timeint:=5000        else
timeint:=TIMEOUTTCPRECEIVE;
      if tickTransfer-source.tick>timeint then begin
       SourceAddFailedAttempt(transfer,source);
       inc(i);
       continue;
      end;
      if not TCPSocket_CanRead(source.socket.socket,0,er) then begin
        if         ((er<>0)         and         (er<>WSAEWOULDBLOCK))         then
SourceAddFailedAttempt(transfer,source);
        inc(i);
        continue;
      end;
      len:=TCPSocket_RecvBuffer(source.socket.socket,@buffer,68,er);
      if er=WSAEWOULDBLOCK then begin
       inc(i);
       continue;
      end;
      if er<>0 then begin
        SourceAddFailedAttempt(transfer,source);
        inc(i);
        continue;
      end;
      SetLength(str,len);
      move(buffer,str[1],len);
      if copy(str,1,20)<>STR_BITTORRENT_PROTOCOL_HANDSHAKE then begin
        SourceAddFailedAttempt(transfer,source);
        inc(i);
        continue;
      end;
      if copy(str,29,20)<>transfer.fhashvalue then begin
       source.status:=btSourceShouldRemove;
       inc(i);
       continue;
      end;
      if length(source.id)=20 then begin
```

```
        if copy(str,49,20)<>source.id then begin
        end;
       end else begin
        source.id:=copy(str,49,20);
       end;
        ParseHandshakeReservedBytes(source,copy(str,21,8));
        source.tick:=tickTransfer;
        SourceSetConnected(source);

        inc(transfer.numConnected);
        //if GetShouldSendBitfield(transfer) then
        if           transfer.fstate<>dlBittorrentMagnetDiscovery           then
SendBitField(transfer,source);
        if source.SupportsExtensions then SendPexHandshake(source);
        if      (source.SupportsDHT)      and      (source.isNotAzureus)      then
SendDHTPort(source);
        globSource:=source;
        globTransfer:=transfer;
//        synchronize(updateVisualGlobsource);
    end;    btSourceweMustSendHandshake:begin //accepted source we received her
handshake, now we send ours
      if (transfer.fsources.count>=50) or
         (transfer.numConnected>15)        then        timeint:=5000        else
timeint:=TIMEOUTTCPRECEIVE;
        if tickTransfer-source.tick>timeint then begin
         source.status:=btSourceShouldRemove;
         inc(i);
         continue;
        end;
        if not TCPSocket_CanWrite(source.socket.socket,0,er) then begin
          if ((er<>0) and (er<>WSAEWOULDBLOCK)) then begin
            source.status:=btSourceShouldRemove;
          end;
          inc(i);
          continue;
        end;
        str:=STR_BITTORRENT_PROTOCOL_HANDSHAKE+
             STR_BITTORRENT_PROTOCOL_EXTENSIONS+
             transfer.fhashvalue+
             thread_bittorrent.mypeerID;
        TCPSocket_SendBuffer(source.socket.socket,pchar(str),length(str),er);
        //ShowMessage('TCPSocket_SendBuffer');
        if er=WSAEWOULDBLOCK then begin
         inc(i);
         continue;
        end;
        if er<>0 then begin
         source.status:=btSourceShouldRemove;
         inc(i);
         continue;
```

```
       end;
       for hi:=0 to transfer.fsources.count-1 do begin
        tmpsource:=transfer.fsources[hi];
        if tmpsource=source then continue;
         if tmpsource.ipC<>source.ipC then continue;
           if              tmpsource.status<>btSourceConnected              then
tmpsource.status:=btSourceShouldRemove
            else source.status:=btSourceShouldRemove;
         Break;//exit;
       end;       source.tick:=tickTransfer;
       SourceSetConnected(source);
       inc(transfer.numConnected);
       GlobTransfer:=transfer;
       globSource:=source;
//       synchronize(updateVisualGlobsource);
       if transfer.fsources.count>BITTORRENT_MAX_ALLOWED_SOURCES then begin
        if  not  DropWorstConnectedInactiveSource(transfer,source,tickTransfer)
then begin
         source.status:=btSourceShouldRemove;
         inc(i);
         continue;
        end;
       end;
       if   transfer.numConnected>BITTORENT_MAXNUMBER_CONNECTION_ACCEPTED   then
begin //limit accepted connections
         if  not  DropWorstConnectedInactiveSource(transfer,source,tickTransfer)
then begin
          source.status:=btSourceShouldRemove;
          inc(i);
          continue;
         end;
       end;
       //if GetShouldSendBitfield(transfer) then
      // if source.SupportsAZmessaging then SendAzHandshake(transfer,source);
        if           transfer.fstate<>dlBittorrentMagnetDiscovery           then
SendBitField(transfer,source);
        if source.SupportsExtensions then SendPexHandshake(source);
        if     (source.SupportsDHT)     and     (source.isNotAzureus)     then
SendDHTPort(source);
    end;
 end; // endof case switch
inc(i);
end;
finally
LeaveCriticalSection(TrackerCriticalSection);
end;
except
end;
if not Terminated then
Sleep(3);
```

```
until Terminated;
end;
procedure tthread_bitTorrent.transferDeal(transfer:TBittorrentTransfer);
var
i,er,len,hi:integer;
source,tmpSource:tbittorrentSource;
str:string;
buffer:array[0..67] of char;
timeint:cardinal;
begin
try
i:=0;
while (i<transfer.fsources.count) do begin
 if terminated then break;
 source:=transfer.fsources[i];
 case source.status of
    btSourceShouldDisconnect:begin
      DisconnectSource(transfer,source,true);
      inc(i);
      continue;
    end;
    btSourceShouldRemove:begin
      transfer.fsources.delete(i);
      RemoveSource(transfer,source);
     continue;
    end;

    btSourceConnected:begin
      while transferDeal(transfer,source) do ;
      inc(i);
      continue;
    end;
    btSourceIdle:begin
      if transfer.fstate=dlPaused then begin
       inc(i);
       continue;
      end;
      if  transfer.numConnected>=BITTORENT_MAXNUMBER_CONNECTION_ESTABLISH  then
begin //no need to connect to more sources
       inc(i);
       continue;
      end;
      if GetNumConnecting(transfer)>=MAX_OUTGOING_ATTEMPTS then begin
       inc(i);
       continue;
      end;
      if                          (source.lastAttempt<>0)                          and
(tick-source.lastAttempt<BTSOURCE_CONN_ATTEMPT_INTERVAL) then begin
       inc(i);
       continue;
```

```
      end;
      if transfer.fErrorCode<>0 then begin
//       Form1.Memo2.Lines.Add('transferDealExit');
       exit;
      end;

      if transfer.isCompleted then
       if source.isSeeder then begin  //this source is a seeder, connect to leechers
only, now that data has been downloaded...
        inc(i);
        continue;
       end;
      source.lastAttempt:=tick;
      if source.socket<>nil then source.socket.free;
      source.ClearOutBuffer;
      source.inbuffer:='';
      source.socket:=TTCPBlockSocket.create(true);
      source.socket.block(false);
      helper_sockets.assign_proxy_settings(source.socket);
      source.tick:=tick;
      source.status:=btSourceConnecting;
      source.IsIncomingConnection:=false;
      source.socket.connect(source.ipS,inttostr(source.port));
      GlobTransfer:=transfer;
      globSource:=source;
//      synchronize(updateVisualGlobsource);
    end;    btSourceConnecting:begin
      if (transfer.fsources.count>=50) or
         (transfer.numConnected>15)        then        timeint:=5000        else
timeint:=TIMEOUTTCPCONNECTION;
      if tick-source.tick>timeint then begin
        SourceAddFailedAttempt(transfer,source);
        inc(i);
        continue;
      end;
      er:=TCPSocket_ISConnected(source.socket);
      if er=WSAEWOULDBLOCK then begin
       inc(i);
       continue;
      end;
      if ((er<>0) and (er<>WSAEWOULDBLOCK)) then begin
        SourceAddFailedAttempt(transfer,source);
        inc(i);
        continue;
      end;
      str:=STR_BITTORRENT_PROTOCOL_HANDSHAKE+
           STR_BITTORRENT_PROTOCOL_EXTENSIONS+
           transfer.fhashvalue+
           thread_bittorrent.mypeerID;
//      ShowMessage(str);
```

```
      TCPSocket_SendBuffer(source.socket.socket,pchar(str),length(str),er);
      if er=WSAEWOULDBLOCK then begin
       inc(i);
       continue;
      end;
      if er<>0 then begin
       SourceAddFailedAttempt(transfer,source);
       inc(i);
       continue;
      end;
      source.status:=btSourceReceivingHandshake;
      source.tick:=tick;
      GlobTransfer:=transfer;
      globSource:=source;
//      synchronize(updateVisualGlobsource);
    end;
    btSourceReceivingHandshake:begin
      if (transfer.fsources.count>=50) or
         (transfer.numConnected>15)        then        timeint:=5000        else
timeint:=TIMEOUTTCPRECEIVE;
      if tick-source.tick>timeint then begin
       SourceAddFailedAttempt(transfer,source);
       inc(i);
       continue;
      end;
      if not TCPSocket_CanRead(source.socket.socket,0,er) then begin
        if         ((er<>0)         and         (er<>WSAEWOULDBLOCK))         then
SourceAddFailedAttempt(transfer,source);
        inc(i);
        continue;
      end;
      len:=TCPSocket_RecvBuffer(source.socket.socket,@buffer,68,er);
      if er=WSAEWOULDBLOCK then begin
       inc(i);
       continue;
      end;
      if er<>0 then begin
        SourceAddFailedAttempt(transfer,source);
        inc(i);
        continue;
      end;
      SetLength(str,len);
      move(buffer,str[1],len);
      if copy(str,1,20)<>STR_BITTORRENT_PROTOCOL_HANDSHAKE then begin
        SourceAddFailedAttempt(transfer,source);
        inc(i);
        continue;
      end;
      if copy(str,29,20)<>transfer.fhashvalue then begin
       source.status:=btSourceShouldRemove;
```

```
        inc(i);
        continue;
       end;
       if length(source.id)=20 then begin
        if copy(str,49,20)<>source.id then begin
        end;
       end else begin
        source.id:=copy(str,49,20);
       end;
        ParseHandshakeReservedBytes(source,copy(str,21,8));
        source.tick:=tick;
        SourceSetConnected(source);

        inc(transfer.numConnected);
        //if GetShouldSendBitfield(transfer) then
        if            transfer.fstate<>dlBittorrentMagnetDiscovery            then
SendBitField(transfer,source);
        if source.SupportsExtensions then SendPexHandshake(source);
        if      (source.SupportsDHT)      and      (source.isNotAzureus)      then
SendDHTPort(source);
        globSource:=source;
        globTransfer:=transfer;
//        synchronize(updateVisualGlobsource);
    end;    btSourceweMustSendHandshake:begin //accepted source we received her
handshake, now we send ours
      if (transfer.fsources.count>=50) or
         (transfer.numConnected>15)        then        timeint:=5000        else
timeint:=TIMEOUTTCPRECEIVE;
        if tick-source.tick>timeint then begin
         source.status:=btSourceShouldRemove;
         inc(i);
         continue;
        end;
        if not TCPSocket_CanWrite(source.socket.socket,0,er) then begin
          if ((er<>0) and (er<>WSAEWOULDBLOCK)) then begin
            source.status:=btSourceShouldRemove;
          end;
          inc(i);
          continue;
        end;
        str:=STR_BITTORRENT_PROTOCOL_HANDSHAKE+
             STR_BITTORRENT_PROTOCOL_EXTENSIONS+
             transfer.fhashvalue+
             thread_bittorrent.mypeerID;
        TCPSocket_SendBuffer(source.socket.socket,pchar(str),length(str),er);
        //ShowMessage('TCPSocket_SendBuffer');
        if er=WSAEWOULDBLOCK then begin
         inc(i);
         continue;
        end;
```

```
       if er<>0 then begin
        source.status:=btSourceShouldRemove;
        inc(i);
        continue;
       end;
       for hi:=0 to transfer.fsources.count-1 do begin
        tmpsource:=transfer.fsources[hi];
        if tmpsource=source then continue;
         if tmpsource.ipC<>source.ipC then continue;
           if              tmpsource.status<>btSourceConnected              then
tmpsource.status:=btSourceShouldRemove
            else source.status:=btSourceShouldRemove;
         exit;
       end;       source.tick:=tick;
       SourceSetConnected(source);
       inc(transfer.numConnected);
       GlobTransfer:=transfer;
       globSource:=source;
//       synchronize(updateVisualGlobsource);
       if transfer.fsources.count>BITTORRENT_MAX_ALLOWED_SOURCES then begin
        if not DropWorstConnectedInactiveSource(transfer,source,tick) then begin
         source.status:=btSourceShouldRemove;
         inc(i);
         continue;
        end;
       end;
       if   transfer.numConnected>BITTORENT_MAXNUMBER_CONNECTION_ACCEPTED   then
begin //limit accepted connections
         if not DropWorstConnectedInactiveSource(transfer,source,tick) then begin
          source.status:=btSourceShouldRemove;
          inc(i);
          continue;
         end;
       end;
       //if GetShouldSendBitfield(transfer) then
      // if source.SupportsAZmessaging then SendAzHandshake(transfer,source);
        if           transfer.fstate<>dlBittorrentMagnetDiscovery           then
SendBitField(transfer,source);
        if source.SupportsExtensions then SendPexHandshake(source);
        if     (source.SupportsDHT)     and     (source.isNotAzureus)     then
SendDHTPort(source);
    end;
 end; // endof case switch
inc(i);
end;
except
end;
end;Procedure SourceSetConnected(source:TBitTorrentSource);
begin
with source do begin
```

```
 Client:=BTIDtoClientName(ID);
 status:=btSourceConnected;
 lastKeepAliveIn:=tick;
 lastKeepAliveOut:=tick;
 isChoked:=true;
 isInterested:=false;
 weArechoked:=true;
 weAreInterested:=false;
 bytes_in_header:=0;
 recv:=0;
 sent:=0;
 bytes_recv_before:=0;
 bytes_sent_before:=0;
 speed_recv:=0;
 speed_send:=0;
 speed_recv_max:=0;
 speed_send_max:=0;
 handshakeTick:=tick;
 lastDataIn:=0;
 lastDataOut:=0;
 snubbed:=false;
 failedConnectionAttempts:=0;
end;
end;
procedure      ParseHandshakeReservedBytes(source:TBittorrentSource;      const
extStr:string);
begin
 with source do begin
  //SupportsAZmessaging:=false;//((ord(extStr[1]) and $80) <> 0);
  SupportsExtensions:=((ord(extStr[6]) and $10) <> 0);
  SupportsFastPeer:=((ord(extStr[8]) and $04) <> 0);
  SupportsDHT:=((ord(extStr[8]) and $01) <> 0);
 end;
end;
procedure CalcNumConnected(transfer:TBitTorrentTransfer);
var
i:integer;
source:TBitTorrentSource;
begin
transfer.numConnected:=0;
for i:=0 to transfer.fsources.count-1 do begin
 source:=transfer.fsources[i];
 if source.status=btSourceConnected then inc(transfer.numConnected);
end;
end;
function GetoptimumNumOutRequests(speedRecv:cardinal):integer;
begin
if speedRecv<KBYTE then result:=1
   else
    if speedRecv<5*KBYTE then result:=2
```

```
     else
      if speedRecv<10*KBYTE then result:=3
       else
        if speedRecv<20*KBYTE then result:=4
         else
          result:=5;
end;
procedure SendBitField(transfer:TBitTorrentTransfer; source:TBitTorrentSource);
var
str:string;
begin
    {
    if source.SupportsFastPeer then begin
         if transfer.isCompleted then begin
          Source_AddOutPacket(source,'',CMD_BITTORRENT_HAVEALL);
          exit;
         end else
           if transfer.fdownloaded=0 then begin
            Source_AddOutPacket(source,'',CMD_BITTORRENT_HAVENONE);
            exit;
           end;
     end;
     }
  str:=transfer.serialize_bitfield;
  source_AddOutPacket(source,str,CMD_BITTORRENT_BITFIELD,true);
end;
procedure TThreadTransfer.SendDHTPort(source:TBitTorrentSource);
var
 portW:word;
 str:string;
begin
 portW:=vars_global.my_mdht_port;
 str:=int_2_word_stringRev(portW);
 source_AddOutPacket(source,str,CMD_BITTORRENT_DHTUDPPORT,true);
end;
procedure tthread_bitTorrent.SendDHTPort(source:TBitTorrentSource);
var
 portW:word;
 str:string;
begin
 portW:=vars_global.my_mdht_port;
 str:=int_2_word_stringRev(portW);
 source_AddOutPacket(source,str,CMD_BITTORRENT_DHTUDPPORT,true);
end;
procedure  TThreadTransfer.SourceAddFailedAttempt(transfer:TBitTorrentTransfer;
source:TBittorrentSource);
begin
 source.socket.free;
 source.socket:=nil;
 source.status:=btSourceIdle;
```

```
 source.inBuffer:='';
 source.bytes_in_header:=0;
 source.ClearOutBuffer;
 inc(source.failedConnectionAttempts);
 if transfer.fsources.count>=100 then begin
  source.status:=btSourceShouldRemove;
  AddBannedIP(transfer,source.ipC);
 end else begin
  if source.failedConnectionAttempts>=BT_MAXSOURCE_FAILED_ATTEMPTS then begin
   source.status:=btSourceShouldRemove;
   AddBannedIP(transfer,source.ipC);
  end;
 end;
 GlobTransfer:=transfer;
 globSource:=source;
// synchronize(updateVisualGlobsource);
end;
procedure
tthread_bitTorrent.SourceAddFailedAttempt(transfer:TBitTorrentTransfer;
source:TBittorrentSource);
begin
 source.socket.free;
 source.socket:=nil;
 source.status:=btSourceIdle;
 source.inBuffer:='';
 source.bytes_in_header:=0;
 source.ClearOutBuffer;
 inc(source.failedConnectionAttempts);
 if transfer.fsources.count>=100 then begin
  source.status:=btSourceShouldRemove;
  AddBannedIP(transfer,source.ipC);
 end else begin
  if source.failedConnectionAttempts>=BT_MAXSOURCE_FAILED_ATTEMPTS then begin
   source.status:=btSourceShouldRemove;
   AddBannedIP(transfer,source.ipC);
  end;
 end;
 GlobTransfer:=transfer;
 globSource:=source;
// synchronize(updateVisualGlobsource);
end;////                    **********                                TRACKER
**********************************************
procedure tthread_bittorrent.checkTracker;
var
i,x:integer;
tran:tbittorrentTransfer;
tracker:tbittorrentTracker;
TrackerReady:Boolean;
begin
 for i:=0 to BitTorrentTransfers.count-1 do begin
```

```
  tran:=BitTorrentTransfers[i];
  TrackerReady:=False;
  for x:=0 to tran.trackers.count-1 do begin
    tracker:=tran.trackers[x];
    if x=tran.trackerIndex then
    begin
      if tracker.Status=bttrackerReadyUpdate then
      begin
        TrackerReady:=True;
        Break;
      end;
    end;
  end;
  for x:=0 to tran.trackers.count-1 do begin
    tracker:=tran.trackers[x];
    if (tick>=tracker.next_poll) and
    //(tracker.Status=bttrackerReadyUpdate) and
    (TrackerReady)
    then
    begin
      tran.trackerIndex:=x;
      //Break;
    end;
  end;
  checkTracker(tran);
 end;
end;
procedure tthread_bitTorrent.checkTracker(transfer:TBittorrentTransfer);
var
 tracker:tbittorrentTracker;
 er:integer;
 sin:TVarSin;
 buffer:array[0..15] of byte;
 action:cardinal;
// localsin:TSockAddrIn;
// lensin:integer;
 HostEnt: PHostEnt;
begin
try
if transfer.fstate=dlPaused then exit;
if transfer.trackers.count=0 then exit;
tracker:=transfer.trackers[transfer.trackerIndex];
if tick<tracker.next_poll then exit;
if not tracker.isudp then begin
 if tracker.socket<>nil then begin
  tracker.socket.free;
  tracker.socket:=nil;
 end;
end else begin
 if tracker.socketUDP<>INVALID_SOCKET then TCPSocket_free(tracker.socketUDP);
```

```
end;
if transfer.fErrorCode<>0 then exit;
if transfer.fstate=dlAllocating then exit;
tracker.ferror:='';
tracker.next_poll:=tick+(tracker.interval*1000)+(30000);
if tracker.isudp then begin
  tracker.UDPtranscationID:=gettickcount;
   FillChar(Sin, Sizeof(Sin), 0);
   Sin.sin_family:=AF_INET;
   Sin.sin_port:=0;
   Sin.sin_addr.s_addr:=0;
   tracker.socketUDP:=synsock.socket(PF_INET,integer(SOCK_DGRAM),IPPROTO_UDP);
   er:=synsock.Bind(tracker.socketUDP,@Sin,SizeOfVarSin(Sin));
   tracker.Status:=bttrackerUDPConnecting;
   tracker.UDPconnectionID:=0;//$41727101980;
   buffer[0]:=0;
   buffer[1]:=0;
   buffer[2]:=4;
   buffer[3]:=$17;
   buffer[4]:=$27;
   buffer[5]:=$10;
   buffer[6]:=$19;
   buffer[7]:=$80;
     action:=0;
    move(action,buffer[8],4);
    move(tracker.UDPtranscationID,buffer[12],4);
    FillChar(UDP_RemoteSin, Sizeof(UDP_RemoteSin), 0);
     UDP_RemoteSin.sin_family:=AF_INET;
     UDP_RemoteSin.sin_port:=synsock.htons(tracker.port);
     UDP_RemoteSin.sin_addr.s_addr:=synsock.inet_addr(PChar(tracker.host));
        if UDP_RemoteSin.sin_addr.s_addr=u_long(INADDR_NONE) then begin
          HostEnt:=synsock.GetHostByName(PChar(tracker.host));
          if HostEnt<>nil then begin

UDP_RemoteSin.sin_addr.s_addr:=u_long(Pu_long(HostEnt^.h_addr_list^)^);
            tracker.host:=ipint_to_dotstring(UDP_RemoteSin.sin_addr.s_addr);
          end;
        end;
   tracker.portW:=UDP_RemoteSin.sin_port;
   tracker.ipC:=UDP_RemoteSin.sin_addr.s_addr;

synsock.SendTo(tracker.socketUDP,buffer,16,0,@UDP_RemoteSin,SizeOf(UDP_RemoteSi
n));

tracker.visualStr:=widestrtoutf8str(AddBoolString(getLangStringW(STR_CONNECTING)
+' ['+tracker.url+']',(not tracker.isScraping)))+
                   widestrtoutf8str(AddBoolString('Scraping
['+GetFullScrapeURL(tracker.url)+']',tracker.isScraping));
tracker.Tick:=tick;
tracker.FError:='';
```

```
end else begin
 tracker.socket:=ttcpblocksocket.create(true);
 tracker.socket.block(false);
 assign_proxy_settings(tracker.socket);
 tracker.socket.Connect(tracker.host,inttostr(tracker.port));
 tracker.Status:=bttrackerConnecting;
end;
tracker.visualStr:=widestrtoutf8str(AddBoolString(getLangStringW(STR_CONNECTING)
+' ['+tracker.url+']',(not tracker.isScraping)))+
                   widestrtoutf8str(AddBoolString('Scraping
['+GetFullScrapeURL(tracker.url)+']',tracker.isScraping));
tracker.Tick:=tick;
tracker.FError:='';
except
end;
end;
procedure tthread_bittorrent.TrackerDeal;
var
i:integer;
tran:TBittorrentTransfer;
begin
 for i:=0 to BitTorrentTransfers.count-1 do begin
  tran:=BittorrentTransfers[i];
    TrackerDeal(tran);
 end;
end;
procedure TThreadTracker.execute;
var
//UDP_RemoteSin:TVarSin;
er,len:integer;
buffer:array[0..1023] of char;
trackerHost,trackerIDStr:string;
stream:tmemorystream;
NumWanted,indexRead:integer;
ind,ind2,contentLength:integer;
contentLengthStr:string;
headerHTTP,OutStr:string;
previous_len:integer;
//tracker:tbittorrentTracker;
UDP_buffer:array[0..16384] of byte;
len_recvd:integer;
action,transactionID,ipC:cardinal;
portW:word;
outudpstr:string;
tracker:tbittorrentTracker;
 //er:integer;
 sin:TVarSin;
 ByteBuffer:array[0..15] of byte;
 //action:cardinal;
// localsin:TSockAddrIn;
```

```
// lensin:integer;
 HostEnt: PHostEnt;
 Bittick:cardinal;
 i:Integer;
 Int64Downloaded : int64;
 Int64Delta : int64;
 Int64Uploaded : int64;
 Url:string;
begin
  FreeOnTerminate:=False;
  if transfer.trackers.count=0 then exit;

  tracker:=transfer.trackers[trackerIndex];
  repeat
  repeat
  i:=1;

try
 EnterCriticalSection(TrackerCriticalSection);
 try
  Bittick:=gettickcount;

if tracker.isudp then begin
 if tracker.socketUDP=INVALID_SOCKET then
 begin
   break;//exit;
 end;
end else begin
 if tracker.socket=nil then
 begin
   break;//exit;
 end;
end;
 case tracker.Status of
   bttrackerUDPConnecting:begin
    if not TCPSocket_canRead(tracker.socketUDP,0,er) then begin
     if Bittick-tracker.Tick>TIMEOUTTCPCONNECTIONTRACKER then begin
       tracker.visualStr:='UDP Error (Timeout ACK1)';
       TCPSocket_Free(tracker.socketUDP);
       //transfer.useNextTracker;
       //tracker.Status:=bttrackerReadyUpdate;
     end;
     break;//exit;
    end;
    len:=SizeOf(UDP_RemoteSin);

len_recvd:=synsock.RecvFrom(tracker.socketUDP,UDP_Buffer,sizeof(UDP_buffer),0,@
UDP_RemoteSin,len);
     if len_recvd<16 then begin
       tracker.visualStr:='UDP Error (Size Error1)';
```

```
        TCPSocket_Free(tracker.socketUDP);
        //transfer.useNextTracker;
        //tracker.Status:=bttrackerReadyUpdate;
       break;//exit;
      end;
      move(UDP_Buffer,action,4);
      if action<>0 then begin
        tracker.visualStr:='UDP Error (Action Mismatch1)';
        TCPSocket_Free(tracker.socketUDP);
        tracker.next_poll:=Bittick+TRACKERINTERVAL_WHENFAILED;
        //transfer.useNextTracker;
        //tracker.Status:=bttrackerReadyUpdate;
        break;//exit;
      end;
      move(UDP_Buffer[4],transactionID,4);
      if tracker.UDPtranscationID<>transactionID then begin
        tracker.visualStr:='UDP Error (ID Mismatch1)';
        TCPSocket_Free(tracker.socketUDP);
        //transfer.useNextTracker;
        //tracker.Status:=bttrackerReadyUpdate;
        break;//exit;
      end;
    end;
    if not Terminated then
    Sleep(1000);
except
end;
  Until Terminated;
  //tracker.Free;
end;
procedure tthread_bitTorrent.TrackerDeal(transfer:tbittorrentTransfer);
var
er,len:integer;
buffer:array[0..1023] of char;
trackerHost,trackerIDStr:string;
stream:tmemorystream;
NumWanted,indexRead:integer;
ind,ind2,contentLength:integer;
contentLengthStr:string;
headerHTTP,OutStr:string;
previous_len:integer;
tracker:tbittorrentTracker;
UDP_buffer:array[0..16384] of byte;
len_recvd:integer;
action,transactionID,ipC:cardinal;
portW:word;
outudpstr:string;
begin
  //ShowMessage('TrackerDeal Started '+transfer.fname+'/'+transfer.fhashvalue);
try
```

```
//Form1.Memo2.Lines.Add('TrackerDeal '+IntToStr(transfer.trackerIndex));
// showmessage(IntToStr(transfer.trackers.count));
if transfer.trackers.count=0 then exit;
//ShowMessage('TrackerDeal Started22 '+transfer.fname+'/'+transfer.fhashvalue);
tracker:=transfer.trackers[transfer.trackerIndex];
//Form1.Memo2.Lines.Add('TrackerDeal '+IntToStr(transfer.trackerIndex));
if tracker.isudp then begin
 if tracker.socketUDP=INVALID_SOCKET then
 begin
   exit;
 end;
end else begin
 if tracker.socket=nil then
 begin
   exit;
 end;
end;
 case tracker.Status of
   bttrackerUDPConnecting:begin
    if not TCPSocket_canRead(tracker.socketUDP,0,er) then begin
     if tick-tracker.Tick>TIMEOUTTCPCONNECTIONTRACKER then begin
       tracker.visualStr:='UDP Error (Timeout ACK1)';
       TCPSocket_Free(tracker.socketUDP);
       transfer.useNextTracker;
       tracker.Status:=bttrackerReadyUpdate;
     end;
     exit;
    end;
    len:=SizeOf(UDP_RemoteSin);

len_recvd:=synsock.RecvFrom(tracker.socketUDP,UDP_Buffer,sizeof(UDP_buffer),0,@
UDP_RemoteSin,len);
if not TCPSocket_CanRead(tracker.socket.socket,0,er) then begin
           if ((er<>0) and (er<>WSAEWOULDBLOCK)) then begin
             tracker.socket.free;
             tracker.socket:=nil;
             tracker.next_poll:=tick+TRACKERINTERVAL_WHENFAILED;
             tracker.visualStr:='Socket Error ('+inttostr(er)+')';
             transfer.useNextTracker;
             tracker.Status:=bttrackerReadyUpdate;
     if len_recvd<16 then begin
       tracker.visualStr:='UDP Error (Size Error1)';
       TCPSocket_Free(tracker.socketUDP);
       transfer.useNextTracker;
       tracker.Status:=bttrackerReadyUpdate;
      exit;
     end;
     move(UDP_Buffer,action,4);
     if action<>0 then begin
       tracker.visualStr:='UDP Error (Action Mismatch1)';
```

```
        TCPSocket_Free(tracker.socketUDP);
        tracker.next_poll:=tick+TRACKERINTERVAL_WHENFAILED;
        transfer.useNextTracker;
        tracker.Status:=bttrackerReadyUpdate;
        exit;
      end;
      move(UDP_Buffer[4],transactionID,4);
      if tracker.UDPtranscationID<>transactionID then begin
        tracker.visualStr:='UDP Error (ID Mismatch1)';
        TCPSocket_Free(tracker.socketUDP);
        transfer.useNextTracker;
        tracker.Status:=bttrackerReadyUpdate;
        exit;
      end;
      move(UDP_Buffer[8],tracker.UDPconnectionID,8);
      tracker.Tick:=tick;
      tracker.Status:=bttrackerUDPReceiving;
      tracker.UDPtranscationID:=gettickcount;
tracker.BufferReceive:='';
        tracker.socket.free;
        tracker.socket:=nil;
         if not tracker.isScraping then begin //it was a regular announce request
          if     transfer.fsources.count>BITTORRENT_MAX_ALLOWED_SOURCES     then
DropOlderIdleSources(transfer);
          tracker.visualStr:=getLangStringW(STR_OK)+AddBoolString('
'+utf8strtowidestr(copy(tracker.WarningMessage,1,100)),length(tracker.WarningMe
ssage)>0);
          tracker.next_poll:=tick+(tracker.interval*1000);

      if ((transfer.isCompleted) and
         (not Tracker.AlreadyCompleted)) then begin
            tracker.UDPevent:=reverseorder(cardinal(1)); //completed
      end else
      if (not tracker.alreadyStarted) then begin
       tracker.UDPevent:=reverseorder(cardinal(2)); // started
      end else begin
        tracker.UDPevent:=0;  //nothing new
      end;
      //ShowMessage('TrackerDeal Started3 '+transfer.fname);

      if       (transfer.fsources.count>=BITTORRENT_DONTASKMORESOURCES)       and
(tracker.UDPevent=0) then begin
       tracker.isScraping:=true;
       outudpstr:=int_2_qword_string(tracker.UDPconnectionID)+
                  chr(0)+chr(0)+chr(0)+chr(2)+//scrape
                  int_2_dword_string(tracker.UDPtranscationID)+
                  transfer.fhashvalue;
       //log('is Scraping '+tracker.URL+': '+outudpstr);
      end else begin
       tracker.isScraping:=false;
```

```
      outudpstr:=int_2_qword_string(tracker.UDPconnectionID)+
                 chr(0)+chr(0)+chr(0)+chr(1)+//announce
                 int_2_dword_string(tracker.UDPtranscationID)+
                 transfer.fhashvalue+
                 mypeerID+
                 int_2_qword_string(reverseorder(int64(transfer.fdownloaded)))+

int_2_qword_string(reverseorder(int64(transfer.fsize-transfer.fdownloaded)))+
                 int_2_qword_string(reverseorder(int64(transfer.fuploaded)))+
                 int_2_dword_string(tracker.UDPevent)+
                 int_2_dword_string(0)+  //ip
                 int_2_dword_string(tracker.UDPKey)+
                 int_2_dword_string(cardinal(-1))+
                 int_2_word_string(vars_global.myport);
      //log('Not is Scraping '+tracker.URL+': '+outudpstr);
     end;
     FillChar(UDP_RemoteSin, Sizeof(UDP_RemoteSin), 0);
     UDP_RemoteSin.sin_family:=AF_INET;
     UDP_RemoteSin.sin_port:=tracker.portw;
     UDP_RemoteSin.sin_addr.s_addr:=tracker.ipC;
     len:=length(outudpstr);
     move(outudpstr[1],buffer,length(outudpstr));

synsock.SendTo(tracker.socketUDP,buffer,len,0,@UDP_RemoteSin,SizeOf(UDP_RemoteS
in));
   end;
   bttrackerUDPReceiving:begin
   if not TCPSocket_canRead(tracker.socketUDP,0,er) then begin
     if tick-tracker.Tick>TIMEOUTTCPCONNECTIONTRACKER then begin
       tracker.visualStr:='UDP Error (Timeout ACK2)';
       TCPSocket_Free(tracker.socketUDP);
       transfer.useNextTracker;
       tracker.Status:=bttrackerReadyUpdate;
     end;
     exit;
    end;
    len:=SizeOf(UDP_RemoteSin);

len_recvd:=synsock.RecvFrom(tracker.socketUDP,UDP_Buffer,sizeof(UDP_buffer),0,@
UDP_RemoteSin,len);
    if len_recvd<8 then begin
       tracker.visualStr:='UDP Error (Size Error2)';
       TCPSocket_Free(tracker.socketUDP);
       transfer.useNextTracker;
       tracker.Status:=bttrackerReadyUpdate;
      exit;
     end;
     move(UDP_Buffer[4],transactionID,4);
     if tracker.UDPtranscationID<>transactionID then begin
       tracker.visualStr:='UDP Error (ID Mismatch2)';
```

```
        TCPSocket_Free(tracker.socketUDP);
        transfer.useNextTracker;
        tracker.Status:=bttrackerReadyUpdate;
        exit;
      end;
      move(UDP_Buffer,action,4);
      if (UDP_Buffer[0]<>0) or
         (UDP_Buffer[1]<>0) or
         (UDP_Buffer[2]<>0) or
         ((UDP_Buffer[3]<>1) and (UDP_Buffer[3]<>2))  then begin
         if UDP_Buffer[3]=3 then begin    // Error
           setLength(tracker.FError,len_recvd-8);
           move(UDP_buffer[8],tracker.FError[1],length(tracker.Ferror));
           TCPSocket_Free(tracker.socketUDP);
            tracker.visualStr:=tracker.FError;
            tracker.next_poll:=tick+TRACKERINTERVAL_WHENFAILED;
            transfer.useNextTracker;
            tracker.Status:=bttrackerReadyUpdate;
            exit;
         end;
        tracker.visualStr:='UDP Error (Action Mismatch2)';
        TCPSocket_Free(tracker.socketUDP);
        transfer.useNextTracker;
        tracker.Status:=bttrackerReadyUpdate;
        exit;
      end;
procedure tthread_download.ReceiveFiles;
var
tempo:cardinal;
tot_amount_recv:integer;
loc_amount_recv:integer;
cicli_da_fare:integer;
begin
if SourcesOnDuty.count=0 then exit;
tempo:=gettickcount;
if download_bandwidth>0 then begin
  if tempo-last_receive_tick<5*TENTHOFSEC then exit;
  last_receive_tick:=tempo;
 tot_amount_recv:=(download_bandwidth*KBYTE) div 2;// due letture al secondo
 loc_amount_recv:=tot_amount_recv div SourcesOnDuty.count;
 if loc_amount_recv>KBYTE then begin  //troppo da inviare per il buffer, riduciamo
  cicli_da_fare:=(loc_amount_recv div KBYTE)+1;
  loc_amount_recv:=loc_amount_recv div cicli_da_fare;
 end else cicli_da_fare:=1;
 ReceiveFiles(loc_amount_recv,cicli_da_fare,Tempo);
end else ReceiveFiles(tempo);
end;
procedure tthread_download.RemoveFromDuty(risorsa:trisorsa_download);
var
download:tdownload;
```

```
ind:integer;
     if (UDP_Buffer[3]=1) and (len_recvd<20) then begin
       tracker.visualStr:='UDP Error (Size Error2)';
       TCPSocket_Free(tracker.socketUDP);
       transfer.useNextTracker;
       tracker.Status:=bttrackerReadyUpdate;
       exit;
     end;
     if (tracker.UDPevent=reverseorder(cardinal(1))) and (transfer.isCompleted)
then tracker.alreadyCompleted:=true else
     if              tracker.UDPevent=reverseorder(cardinal(2))              then
tracker.alreadyStarted:=true;
    if UDP_Buffer[3]=1 then begin  //announcing?
     move(UDP_Buffer[8],tracker.Interval,4);
      tracker.interval:=reverseorder(tracker.interval);
     move(UDP_Buffer[12],tracker.Leechers,4);
      tracker.leechers:=reverseorder(tracker.leechers);
     move(UDP_Buffer[16],tracker.Seeders,4);
      tracker.seeders:=reverseorder(tracker.seeders);
      indexread:=20;
      while (indexRead<len_recvd) do begin
       move(UDP_buffer[indexRead],ipC,4);
       move(UDP_buffer[indexRead+4],portW,2);
       transfer.addSource(ipC,reverseorder(portW),'','UDP');
       //ShowMessage('UDP: '+IntToStr(ipC));
       inc(indexRead,6);
      end;

     end else begin   //scraping?
       if len_recvd>=20 then begin
        move(UDP_Buffer[8],tracker.seeders,4);
        move(UDP_Buffer[16],tracker.leechers,4);
       end;
     end;
     TCPSocket_Free(tracker.socketUDP);
     tracker.next_poll:=tick+(tracker.interval*1000);
     tracker.visualStr:=getLangStringW(STR_OK);
     transfer.useNextTracker;
     tracker.Status:=bttrackerReadyUpdate;
     exit;
   end;
   bttrackerConnecting:begin
      if tick-tracker.Tick>TIMEOUTTCPCONNECTIONTRACKER then begin
       tracker.socket.free;
       tracker.socket:=nil;
       tracker.next_poll:=tick+TRACKERINTERVAL_WHENFAILED;
       tracker.visualStr:='Socket Error (Timeout)';
       transfer.useNextTracker;
       tracker.Status:=bttrackerReadyUpdate;
       exit;
```

```
      end;
      er:=TCPSocket_ISConnected(tracker.socket);
      if er=WSAEWOULDBLOCK then exit;
      if ((er<>0) and (er<>WSAEWOULDBLOCK)) then begin
        tracker.socket.free;
        tracker.socket:=nil;
        tracker.next_poll:=tick+TRACKERINTERVAL_WHENFAILED;
        tracker.visualStr:='Socket Error ('+inttostr(er)+')';
        transfer.useNextTracker;
        tracker.Status:=bttrackerReadyUpdate;
        exit;
      end;
      if                          tracker.port<>80                          then
trackerHost:=tracker.host+':'+inttostr(tracker.port)
       else trackerHost:=tracker.host;
   if tracker.isScraping then begin
   if Pos('=',tracker.Url)>0 then
   OutStr:='GET '+GetScrapePathFromUrl(tracker.Url)+'&'+
           'info_hash='+fullUrlEncode(transfer.fhashvalue)+
           ' HTTP/1.1'+CRLF+
           'User-Agent: '+const_ares.AGENT_NAME+' '+versioneares+CRLF+
           'Connection: close'+CRLF+
           'Host: '+trackerHost+CRLF+
           'Accept: text/html, */*'+CRLF+CRLF
   else
   OutStr:='GET '+GetScrapePathFromUrl(tracker.Url)+'?'+
           'info_hash='+fullUrlEncode(transfer.fhashvalue)+
           ' HTTP/1.1'+CRLF+
           'User-Agent: '+const_ares.AGENT_NAME+' '+versioneares+CRLF+
           'Connection: close'+CRLF+
           'Host: '+trackerHost+CRLF+
           'Accept: text/html, */*'+CRLF+CRLF;
           //OutStr:=StringReplace(OutStr,  '.php&',  '.php?',  [rfReplaceAll,
rfIgnoreCase]);
           //OutStr:=StringReplace(OutStr, 'scrape&', 'scrape?', [rfReplaceAll,
rfIgnoreCase]);
   end else begin
       NumWanted:=TRACKER_NUMPEER_REQUESTED;
       if ((transfer.isCompleted) and
           (not Tracker.AlreadyCompleted)) then begin
           tracker.CurrTrackerEvent:='&event=completed';
       end else
        if ((tracker.alreadyStarted) or
            (tracker.alreadyCompleted)) then tracker.CurrTrackerEvent:=''
         else
          tracker.CurrTrackerEvent:='&event=started';
         if    transfer.fsources.count>=BITTORRENT_MAX_ALLOWED_SOURCES    then
NumWanted:=0;

      if                        tracker.trackerID<>''                       then
```

```
trackerIDStr:='&trackerid='+tracker.trackerID
       else trackerIDStr:='';
   if Pos('=',tracker.Url)>0 then
   OutStr:='GET '+GetPathFromUrl(tracker.Url)+'&'+
           'info_hash='+fullUrlEncode(transfer.fhashvalue)+
           '&peer_id='+thread_bittorrent.mypeerID+
           trackerIDStr+
           '&port='+inttostr(vars_global.myport)+
           '&uploaded='+inttostr(transfer.fuploaded)+
           '&downloaded='+inttostr(transfer.fdownloaded)+
           '&left='+inttostr(transfer.fsize-transfer.fdownloaded)+
           tracker.CurrTrackerEvent+
           '&numwant='+inttostr(NumWanted)+
           '&compact=1'+
           '&key='+thread_bittorrent.myrandkey+
           ' HTTP/1.1'+CRLF+
           'User-Agent: '+const_ares.AGENT_NAME+' '+versioneares+CRLF+
           'Connection: close'+CRLF+
           'Host: '+trackerHost+CRLF+
           'Accept: text/html, */*'+CRLF+CRLF
   else
   OutStr:='GET '+GetPathFromUrl(tracker.Url)+'?'+
           'info_hash='+fullUrlEncode(transfer.fhashvalue)+
           '&peer_id='+thread_bittorrent.mypeerID+
           trackerIDStr+
           '&port='+inttostr(vars_global.myport)+
           '&uploaded='+inttostr(transfer.fuploaded)+
           '&downloaded='+inttostr(transfer.fdownloaded)+
           '&left='+inttostr(transfer.fsize-transfer.fdownloaded)+
           tracker.CurrTrackerEvent+
           '&numwant='+inttostr(NumWanted)+
           '&compact=1'+
           '&key='+thread_bittorrent.myrandkey+
           ' HTTP/1.1'+CRLF+
           'User-Agent: '+const_ares.AGENT_NAME+' '+versioneares+CRLF+
           'Connection: close'+CRLF+
           'Host: '+trackerHost+CRLF+
           'Accept: text/html, */*'+CRLF+CRLF;
           //OutStr:=StringReplace(OutStr,  '.php&',  '.php?',  [rfReplaceAll,
rfIgnoreCase]);
           //OutStr:=StringReplace(OutStr,       'announce&',       'announce?',
[rfReplaceAll, rfIgnoreCase]);
     end;
TCPSocket_SendBuffer(tracker.socket.socket,pchar(OutStr),length(OutStr),er);
        if er=WSAEWOULDBLOCK then begin
         exit;
        end;
function               tthread_download.DoIdleSlowestSource(download:tdownload;
risorsa:trisorsa_download):boolean;
var
```

```
i:integer;
ris:trisorsa_download;
tempo:cardinal;
begin
result:=false;
for i:=0 to download.lista_risorse.count-1 do begin
 ris:=download.lista_risorse[i];
 if ris=risorsa then continue;
 if ris.piece=nil then continue;//should never happen
  if isSourceState(ris,srs_connecting) then begin
   SourceDoIdle(ris,true);
   result:=true;
   exit;
  end;
end;if                    download.lista_risorse.count>1                    then
download.lista_risorse.sort(ordina_risorse_slower_prima);
tempo:=gettickcount;
for i:=0 to download.lista_risorse.count-1 do begin
 ris:=download.lista_risorse[i];
 if ris=risorsa then continue;
  if ris.state<>srs_receiving then continue;
   if tempo-ris.started_time<5000 then continue;   // started soon
    if ris.speed>(risorsa.speed div 2) then continue; // reasonably fast
     if ris.size_to_receive<(ris.speed * 5) then continue; //...going to be
completed within 5 seconds
         RemoveFromDuty(ris);
         update_hole_table(download);
         SourceDoIdle(ris,true);
           result:=true;
           exit;
end;
end;
        if er<>0 then begin
          tracker.socket.free;
          tracker.socket:=nil;
          tracker.next_poll:=tick+TRACKERINTERVAL_WHENFAILED;
          tracker.visualStr:='Socket Error ('+inttostr(er)+')';
          transfer.useNextTracker;
          tracker.Status:=bttrackerReadyUpdate;
          exit;
        end;
       tracker.BufferReceive:='';
       tracker.Tick:=tick;
       tracker.Status:=bttrackerReceiving;
   end;
   bttrackerReceiving:begin
         if tick-tracker.Tick>TIMEOUTTCPRECEIVETRACKER then begin
          tracker.socket.free;
          tracker.socket:=nil;
          tracker.next_poll:=tick+TRACKERINTERVAL_WHENFAILED;
```

```
          tracker.visualStr:='Socket Error (Timeout)';
          transfer.useNextTracker;
          tracker.Status:=bttrackerReadyUpdate;
          exit;
        end;
        if not TCPSocket_CanRead(tracker.socket.socket,0,er) then begin
          if ((er<>0) and (er<>WSAEWOULDBLOCK)) then begin
            tracker.socket.free;
            tracker.socket:=nil;
            tracker.next_poll:=tick+TRACKERINTERVAL_WHENFAILED;
            tracker.visualStr:='Socket Error ('+inttostr(er)+')';
            transfer.useNextTracker;
            tracker.Status:=bttrackerReadyUpdate;
          end;
          exit;
        end;

len:=TCPSocket_RecvBuffer(tracker.socket.socket,@buffer,sizeof(buffer),er);
          if er=WSAEWOULDBLOCK then exit;
          if er<>0 then begin
            tracker.socket.free;
            tracker.socket:=nil;
            tracker.next_poll:=tick+TRACKERINTERVAL_WHENFAILED;
            tracker.visualStr:='Socket Error ('+inttostr(er)+')';
            transfer.useNextTracker;
            tracker.Status:=bttrackerReadyUpdate;
            exit;
          end;
          if len=0 then begin
            tracker.socket.free;
            tracker.socket:=nil;
            tracker.next_poll:=tick+TRACKERINTERVAL_WHENFAILED;
            tracker.visualStr:='Socket Error';
            transfer.useNextTracker;
            tracker.Status:=bttrackerReadyUpdate;
            exit;
          end;
        tracker.Tick:=tick;

        previous_len:=length(tracker.BufferReceive);
        Setlength(tracker.BufferReceive,previous_len+len);
        move(buffer,tracker.BufferReceive[previous_len+1],len);
        ind:=pos(CRLF+CRLF,tracker.BufferReceive);
        if ind>0 then begin
          headerHTTP:=copy(tracker.BufferReceive,1,ind-1);
ind2:=pos('content-length:',lowercase(headerHTTP));
          if ind2>0 then begin   // do we have 'Content-Length:' ?
            contentLengthStr:=copy(headerHttp,ind2+15,length(headerHTTP));

contentLengthStr:=trim(copy(contentLengthStr,1,pos(CRLF,contentLengthStr)-1));
```

```
            contentLength:=strtointDef(contentLengthStr,0);
              if contentLength+length(headerHttp)>length(tracker.BufferReceive)
then begin// not enough
                exit;
              end;
          end;
          delete(tracker.BufferReceive,1,ind+3);
        end else begin
        if     ((pos('HTTP',tracker.BufferReceive)=1)     and     (pos('     200
OK'+chr(10),tracker.BufferReceive)>0)) then begin

delete(tracker.BufferReceive,1,pos(chr(10)+chr(10),tracker.BufferReceive)+1);
        end;
        end;
      if length(tracker.BufferReceive)>0 then begin
        stream:=tmemorystream.create;
         stream.Write(tracker.BufferReceive[1],length(tracker.BufferReceive));
         stream.position:=0;
         if not tracker.isScraping then tracker.Load(stream)
          else tracker.parseScrape(stream);
        stream.free;
      end;
      tracker.BufferReceive:='';
      tracker.socket.free;
      tracker.socket:=nil;
       if not tracker.isScraping then begin //it was a regular announce request
        if     transfer.fsources.count>BITTORRENT_MAX_ALLOWED_SOURCES     then
DropOlderIdleSources(transfer);
        tracker.visualStr:=getLangStringW(STR_OK)+AddBoolString('
'+utf8strtowidestr(copy(tracker.WarningMessage,1,100)),length(tracker.WarningMe
ssage)>0);
        tracker.next_poll:=tick+(tracker.interval*1000);

        if length(tracker.FError)>0 then begin
          tracker.visualStr:=tracker.FError;


         end else begin
           if          tracker.CurrTrackerEvent='&event=started'          then
tracker.alreadyStarted:=true
            else
             if        tracker.CurrTrackerEvent='&event=completed'        then
tracker.alreadyCompleted:=true;
           if ((tracker.seeders=0) and
               (tracker.leechers=0) and
               (tracker.SupportScrape)) then begin
                          tracker.isScraping:=true;
                          tracker.next_poll:=0;
                          end;
         end;
```

```
          transfer.useNextTracker;
          tracker.Status:=bttrackerReadyUpdate;
        end else begin  // was scraping....
         tracker.visualStr:=getLangStringW(STR_OK)+AddBoolString('
'+utf8strtowidestr(copy(tracker.WarningMessage,1,100)),length(tracker.WarningMe
ssage)>0);
         tracker.isScraping:=false;
         tracker.next_poll:=tick+(tracker.interval*1000);
          if length(tracker.FError)>0 then tracker.visualStr:=tracker.FError;
         transfer.useNextTracker;
         tracker.Status:=bttrackerReadyUpdate;
        end;   end;
 end;
 except
 end;
end;
procedure DropOlderIdleSources(transfer:TBitTorrentTransfer);
var
i:integer;
source:TBitTorrentSource;
begin
transfer.fsources.sort(SortSourcesOlderFirst);
i:=0;while ((i<transfer.fsources.count) and
       (transfer.fsources.count>BITTORRENT_MAX_ALLOWED_SOURCES)) do begin
 source:=transfer.fsources[i];
 if source.status<>btSourceIdle then begin
  inc(i);
  continue;
 end;
 if source.handshakeTick=0 then begin //leave room for newest sources
  inc(i);
  continue;
 end;
 source.status:=btSourceShouldRemove;
 inc(i);
end;
end;
procedure TThreadTracker.log(txt:string);
begin
outputdebugstring(pchar(formatdatetime('hh:nn:ss',now)+'> '+txt));
end;
procedure TThreadTransfer.log(txt:string);
begin
outputdebugstring(pchar(formatdatetime('hh:nn:ss',now)+'> '+txt));
end;
procedure tthread_bitTorrent.log(txt:string);
begin
outputdebugstring(pchar(formatdatetime('hh:nn:ss',now)+'> '+txt));
end;
end.
```