MPA MPA update (-03) (Marker PDU Aligned Framing for TCP)

Paul R. Culley

HP

6-26-2003

Changes

- Allow optional receiver specified Markers
- Allow optional, mutually agreed, CRC disabling
- Require that MPA send and receive a streaming mode “Key” that validates the ULP parameter startup, prior to sending first FPDU.
  - MPA key also declares: CRC usage, Marker usage
- Modified startup section to deal with “Start Key”
- Added informational sections on:
  - Effects on TCP of MPA
  - Private data exchange mechanism

Consensus: Markers

- Make marker usage OPTIONAL
- put control of this option entirely in the hands of the receiver
- senders MUST implement both markers and not-markers
- receivers MUST implement one OR the other
  - each receiver gets to pick which one is used as the benefits of markers are primarily on the receiver side.
- With the following additions:
  - When markers are used, the marker interval MUST be 512 bytes.

Consensus: CRCs

(A) CRC support MUST be implemented.

(B) CRC usage is all-or-nothing on header and data.

- There is no requirement for CRC on header but not on data or CRC on data but not on header.

(C) CRC usage is bidirectional.

- Either CRCs are used in both directions on a connection, or in neither. There is no requirement for CRC usage in one direction and not the other.

(D) CRC omission requires consent of both ends of the connection.

- If either end of the connection specifies that CRCs are to be used, then they MUST be used [in both directions (C)].

(E) CRCs are OPTIONAL to use, but (per (D) above)

- either end of the connection can force CRC usage in both directions.

(E') CRC usage is an administrative configuration decision

- it is not intended to be visible to ULPs.

Author's Proposals

- When Markers are used, 1st Marker is 32-bit zero, inserted at start point of TCP stream.
  - No padding or adjusting to make them Modulo TCP Sequence number
    - Not possible in fully layered sender implementation
- CRC field is always present
  - when CRC mode is disabled, field may contain anything, is not checked.

Entering MPA/DDP/RDMA mode

From WG reflector, potential startup scenarios:

(1) Enter immediately on startup, possibly with an initial frame containing “private data” required to setup the connection.

(2) Enter at some subsequent point, "quiescing" the connection first.

(3) Enter at some subsequent point, on the fly without a hiccup.

Author’s proposals

- Private data is carried in TCP stream mode prior to MPA enablement
- To simplify implementer’s job, make more likely to get it right, only support (1,2) above.
  - Enter MPA after some potential streaming data has been sent, "quiescing" the connection first
  - Minimum streaming data is 0 bytes
  - Allows easier “Stack” switch
    - no more receive data expected at switch point avoids ULP buffer grab problems

Author's proposals (cont.)

- MPA 128 bit "Start Key" is a function of:
  - MPA operating parameters (CRC usage, Marker usage)
  - A fixed value key (text: "MPA ident frame")
  - Contains a Rev level and reserved bits
- If received "key" does not match the expected value, the connection is closed.
  - Provides a cross check that ULP setup makes sense.
- "Start Key" may not be the same in each direction
  - MPA marker mode or CRC usage may be different

Author's Proposals (cont)

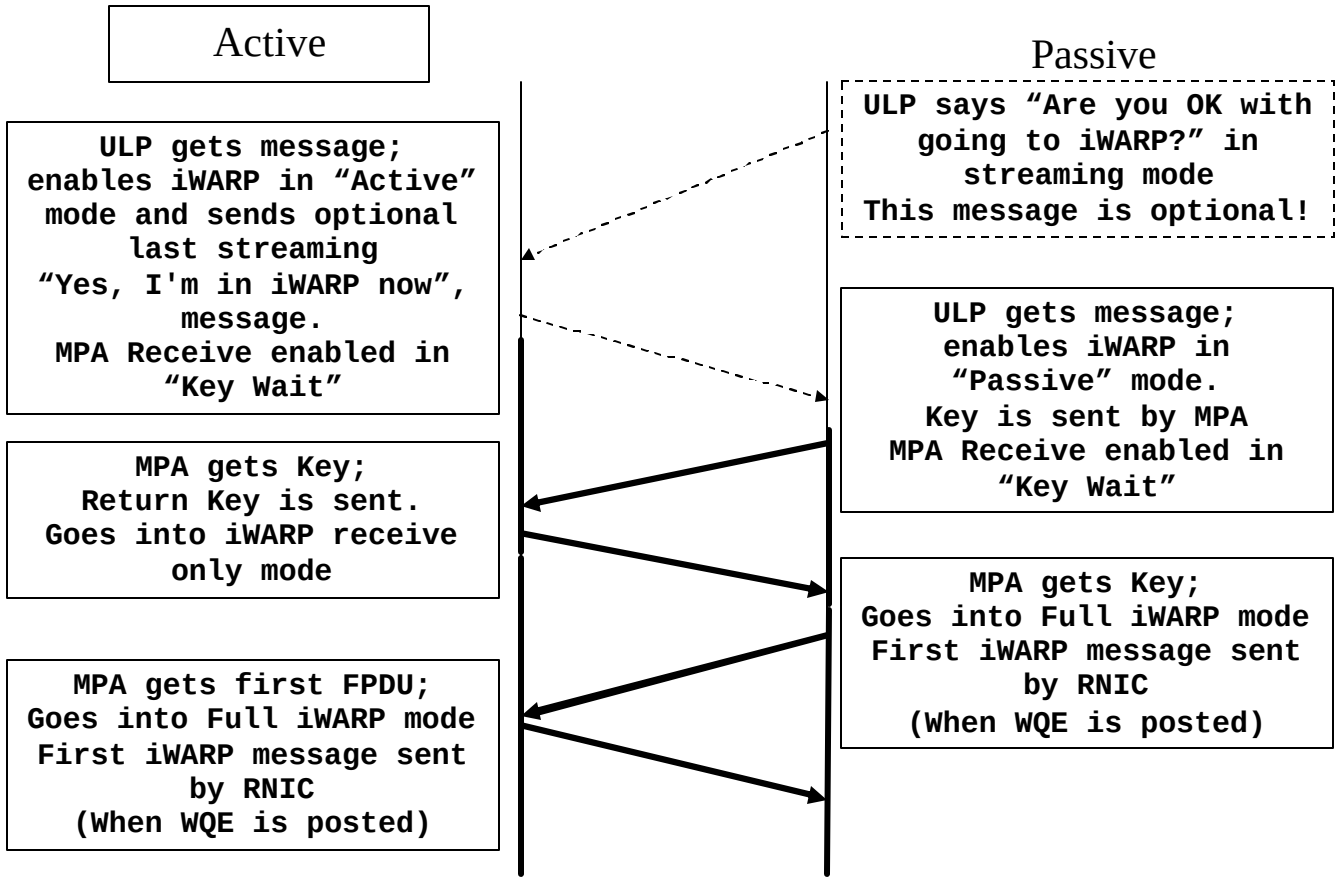
- When MPA is started in the "Passive"* mode, the MPA implementation MUST send a valid "Start Key".
- When MPA is started in the "Active"* mode, the MPA implementation MUST wait until a "Start Key" is received. After the received "Start Key" is validated, the MPA implementation MUST send a valid "Start Key".
- MPA implementations MUST receive and validate a "Start Key" before starting to interpret the data received as FPDUs and passing any received ULPDUs to DDP.

* The choice of the "Active" or "Passive" nomenclature is arbitrary; the "Active" side is the one that sends the last streaming data.

Author’s Proposals (cont)

- MPA "Passive" mode implementations MUST receive and validate a "Start Key" before sending any FPDUs or markers.
- MPA "Active" mode implementations MUST receive and validate at least one FPDU before sending any FPDUs or markers.

IWarp Startup sequence



Start Key

Key (“MPA ident frame”)	M	C	Res(0)	Rev (0)
120 b (15 ASCII characters)	1b	1b	2b	4b

Key: The 15 char text string “MPA framing mode”

M: Marker bit (0=No marker, 1=Marker)

C: CRC bit (0=No CRC desired, 1=CRC required)

Rev: MPA revision (0 for this version)

Res: Reserved transmit as zero, not checked

Faq

- Why quiesce before enabling MPA/DDP?
  - Eliminates confusion on transition; ULP can completely empty receive buffer; does not have to worry about getting too much (part of “start key”) or too little (“junk” before “start key”).
- Why must passive side wait for “start key” before sending FPDU?
  - Ensures that BOTH ends are MPA/DDP endpoints before lots of data are sent.
    - only the “Start key” would be received by a non-MPA endpoint
  - MPA knows about marker and CRC modes
    - So first (few) FPDUs are not different (with Marker/CRC)

Faq

- Why can't Active side send FPDUs as soon as "Start Key" is received?
  - Allows simpler "passive" side receiver design; "Start Key" and first FPDU cannot end up in same segment; allows time after Start Key arrives to reconfigure for FPDU reception.
- Why keep CRC field when no CRCs are used?
  - Allows simpler transmitters that always insert CRCs
- Why keep first all zero marker?
  - Allows compatibility with designs in progress

Handling Private Data

- Non-normative proposal
- Used for “immediate” startup in iWARP
- When used, must be expected by ULP
  - Not specified by MPA, but probably uses port number or locator protocol.
- One Streaming mode message
  - per direction, ULP needs to specify which direction, and if both directions.
- Consists of two byte length, followed by “private data”

Faq

- Why not include “private data” in the “Start key”?
  - Doing so adds another mode and application interface to MPA
    - During “Key mode”, MPA would not be associated with a QP, must have a mechanism to pass received private data to app, would need an additional ULP call to go to full MPA/DDP mode with QP.
  - Can be done adequately by using streaming mode interfaces (sockets).

Faq

- How are Active/Active Peer to Peer connections dealt with?
  - If Peer to Peer connections are used, they MUST use some ULP mechanism to identify the MPA/DDP "Active" or "Passive" sides.
  - Can be based on Unique endpoint name, or
  - Can be some GUID in "private data", or
  - Can be a full streaming negotiation

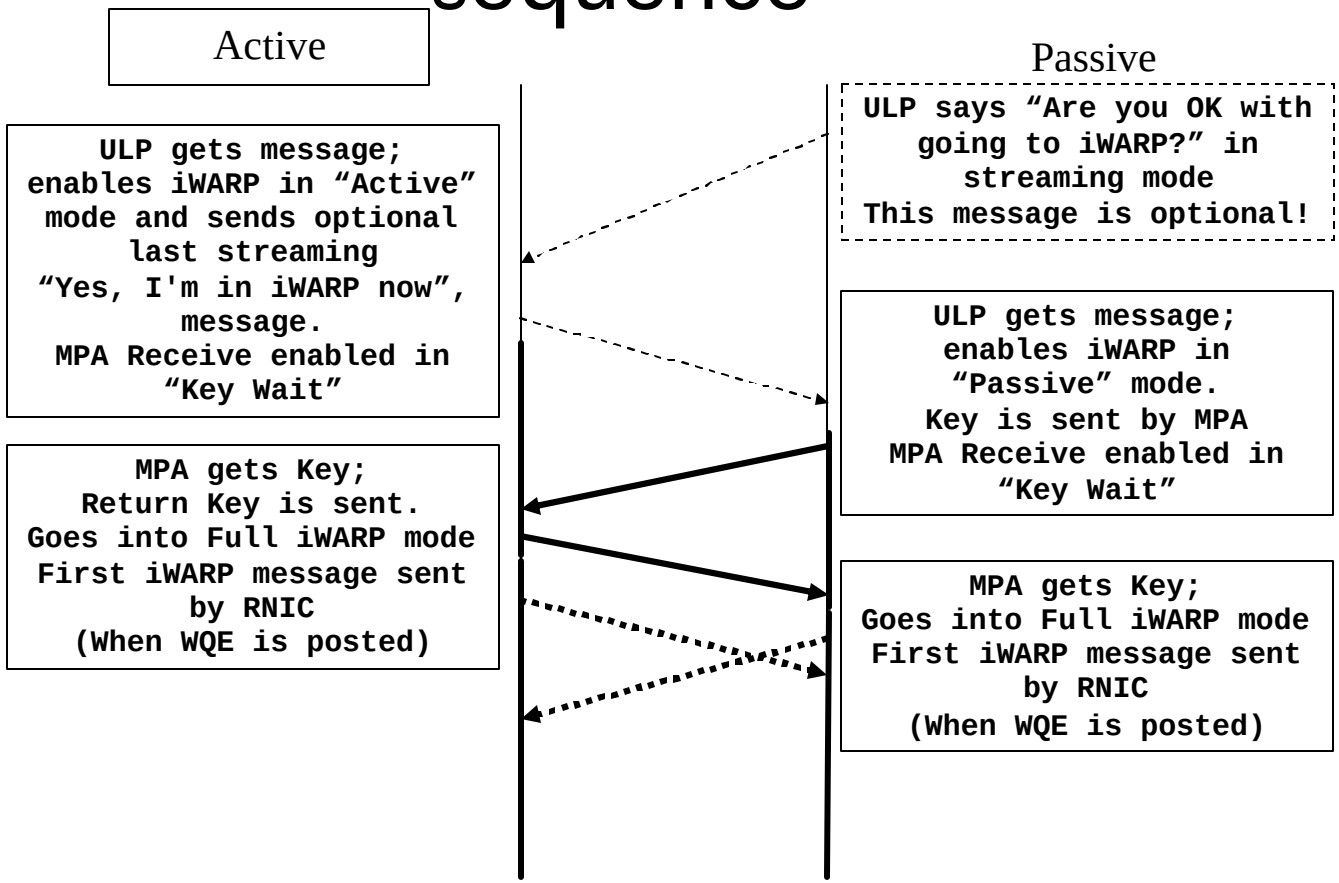
Issues

- Considerable dissention on handling “private data”
  - Some want it incorporated into MPA “Start Key”, despite extra ULP interface required.
    - Shorter startup sequence, by one round trip, when private data is needed.
  - Some want entire “Start Key” and “Private data” left up to ULP in streaming mode.
    - Simplifies startup sequence for RNIC implementers considerably
    - Potentially shorter startup sequence, by one round trip, when private data is needed, or when streaming messages are sent prior to iWARP start.

Issues

- Need to discuss tradeoffs between simpler implementations, and startup speed.
  - Could allow “Start Key” to be sent immediately after last streaming message is sent (by same side).
    - Complicates “Stack” handoff
  - Could allow FPDU to be sent as soon as Start Key is sent.
    - Don’t know Marker/CRC mode yet
  - Could allow FPDUs to be sent immediately after “Start Key” reception on both sides.
    - Complicates FPDU reception logic

Alternative IWarp Startup sequence

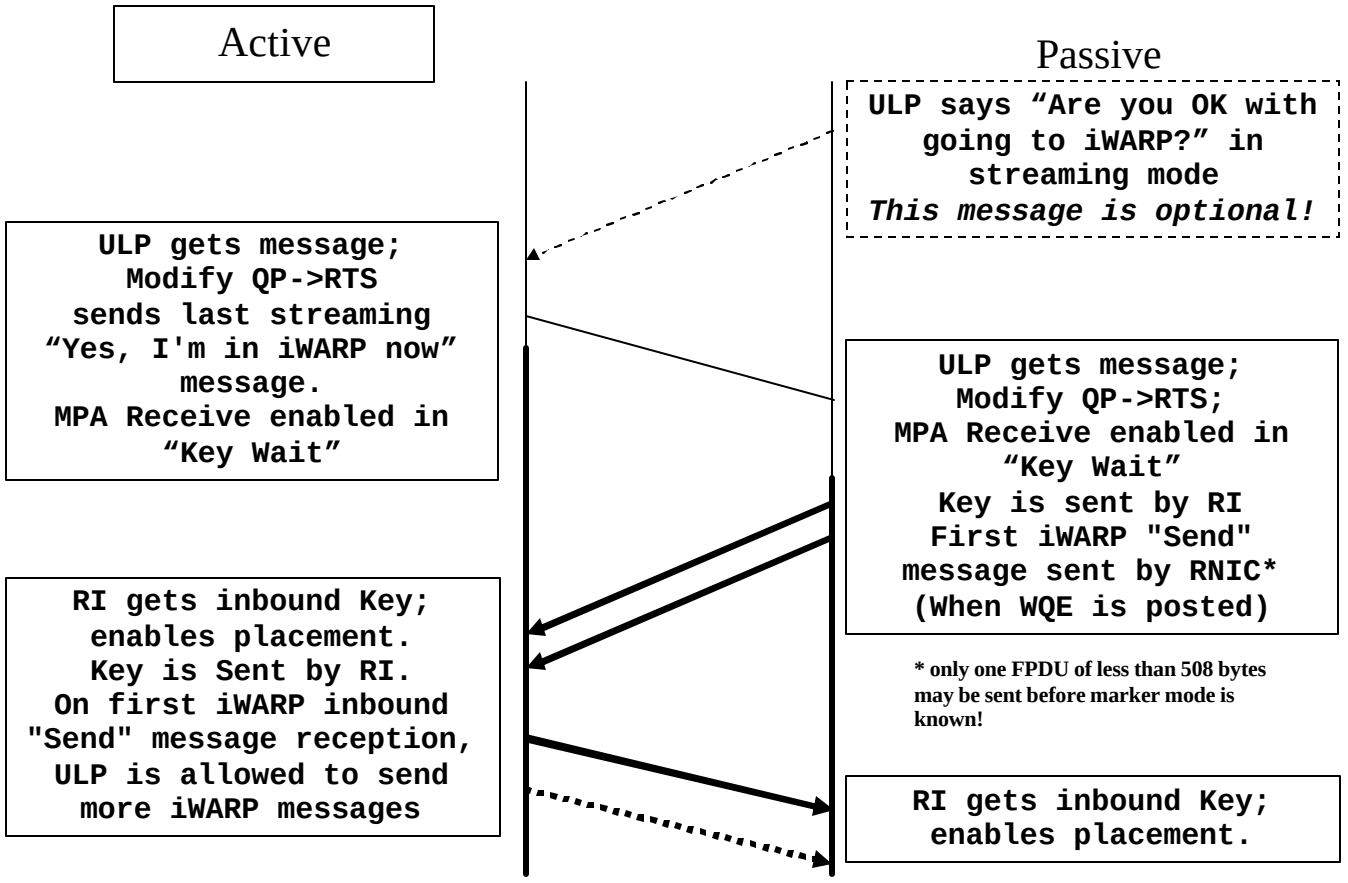


End

Backup slides

- To be shown only if discussion makes this desirable.

IWarp Startup sequence (A)



Sequence (A)

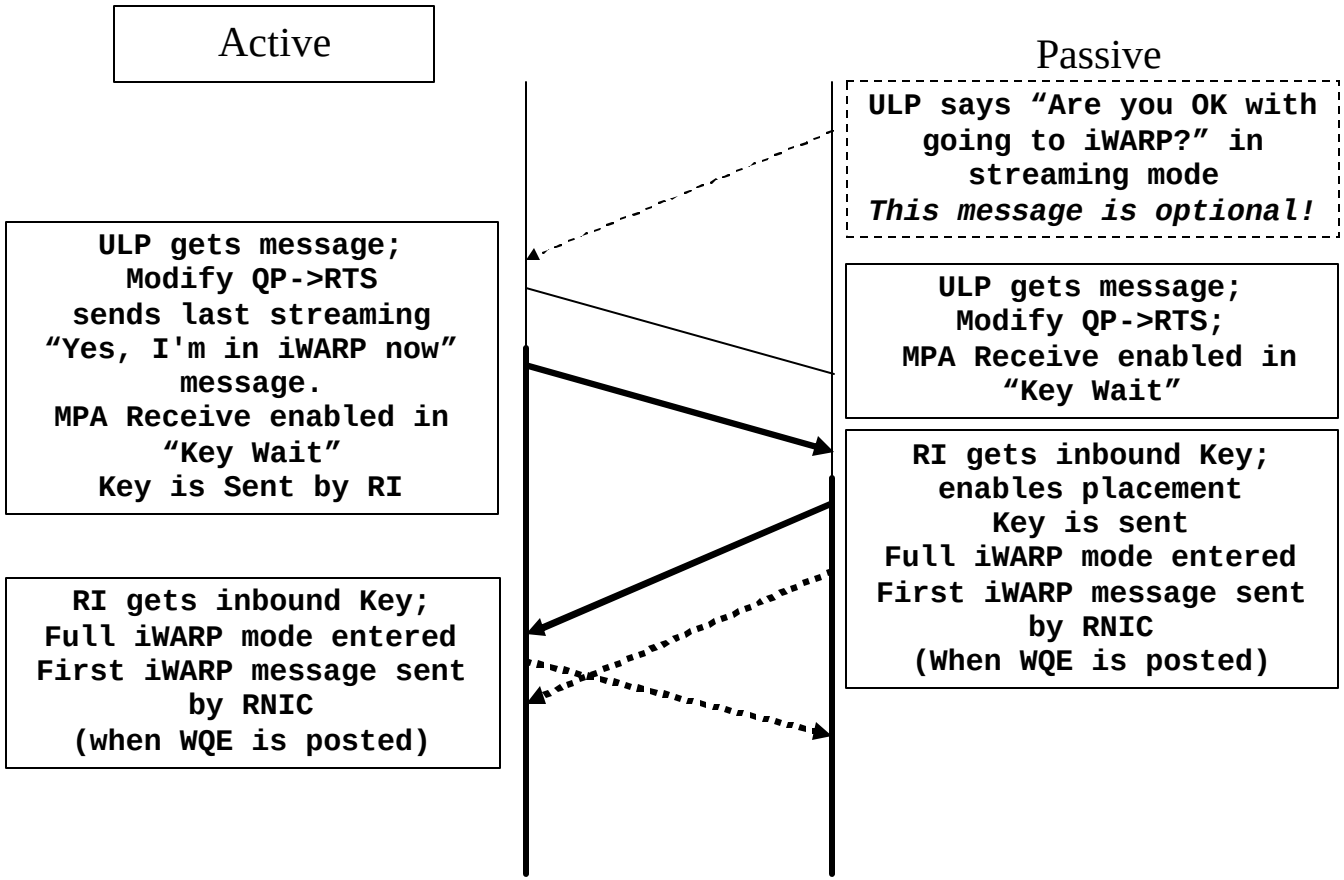
- Key is sent just prior to first FPDU transmission.
- Active side Races:
  - RI must be prepared for rapid return of Key and FPDU, once stream message leaves NIC (so must get into iWARP reception before it can arrive). *Same race as described in Verbs…*
  - Key (or its remainder) may arrive in same TCP segment as first FPDU. RI must manage validation and transition to full placement without losing FPDU.
- Passive side races:
  - Key (or its remainder) may arrive in same TCP segment as first FPDU. RI must manage validation and transition to full placement without losing FPDU.

Implementation notes (A)

- Interlocks:
  - For Active side, after sending last stream message, the RI MUST wait until receiving a valid remote Key before going to full RTS. When the key arrives it MUST send its key. It may send any available FPDUs after sending key.
  - For Passive side, the RI MUST transmit the Key on transition into RTS state, and may transmit (limited*) FPDUs, but waits for incoming Key before going to full iWARP.

*Limited to 508 bytes total

IWarp Startup sequence (B)



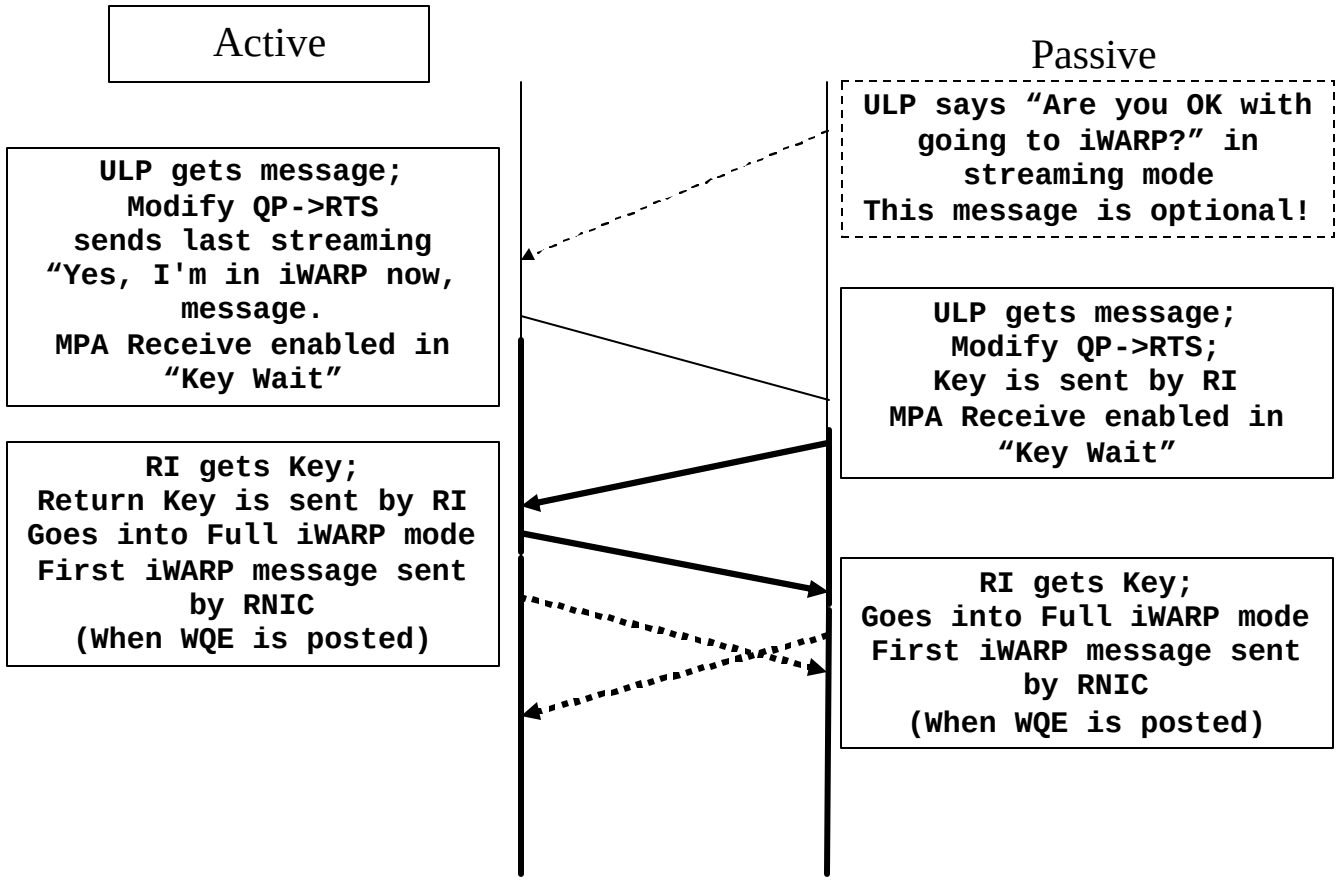
Sequence (B)

- Key is sent when QP is enabled for iWARP mode.
- Active side Races:
  - RI must be prepared for rapid return of key and FPDU, once stream message leaves NIC (so must get into iWARP reception before it can arrive). *Same race as described in Verbs…*
  - Key (or its remainder) may arrive in same TCP segment as first FPDU. RI must manage validation and transition to full placement without losing FPDU.
- Passive side races:
  - Last ULP stream message and Startup Key may arrive in the same TCP segment.
    - Receiving ULP MUST not remove Key from stream buffer before enabling iWARP mode.

Implementation notes (B)

- Interlocks:
  - After sending last stream message, the Active side MUST send Key on entry to RTS, and wait until key is received before transmitting any available FPDUs.
  - Passive side MUST wait for key on entry to RTS, when key arrives it MUST send key. It may send FPDUs after sending key.

IWarp Startup sequence (C)



Sequence (C)

- Key exchange is done when QP is enabled for iWARP mode.
  - Exchange adds round trip to latency for passive side!!!
  - Reduces races to Active side only:
    - RI must be prepared for rapid return of Key, once stream message leaves NIC (so must get into iWARP "awaiting Key" mode before it can arrive).
    - RI must be prepared for rapid return of first FPDU, once Key leaves NIC (so must get into Full iWARP reception mode before it can arrive).
  - Active side must deal with key retransmissions in iWARP mode

‘C’ Interlocks

- Interlocks:
  - For Active side, after sending last stream message, the RI MUST wait until receiving the valid remote Key and then transmit its Key after which it can go to full RTS and transmit FPDUs, as available.
  - For Passive side, the RI MUST transmit the Key on transition into RTS state, and MUST wait for a valid incoming Key before going to full iWARP and transmitting available FPDUs.

Comparison

A) Pro: Separation between streaming mode and “Key” is clear, No special requirements on ULP

Pro: No new round trips required.

Con: Active side must delay “Key” transmission until “Key” or first FPDU reception

Con: Only one FPDU can be sent and it must be limited to 508 bytes or less.

B) Pro: Sending Key when transiting out of Idle seems clearer; same on both sides.

Pro: No new round trips required.

Pro: No limitation on length/number of first FPDUs

Con: ULP must take care not to remove the “Key” from the streaming buffer before transitioning to RTS.

C) Pro: Separation between streaming mode and “Key” is clear, No special requirements on ULP

Pro: no new races

- (just the usual initial Stream out to iWARP in race)

Con: Adds a round trip message pair before passive side can send FPDU

Challenge/Response keys

Ch:

"MPA framing mode"	Cookie	M	C	Rev	Reserved (0)
128 b	32 b	1b	1b	6b	24b

Resp:

RSA MD5 Hash	M	C	Rev	Reserved (0)
128 b	1b	1b	6b	24b

1: First to send includes a Random Cookie

2: Responder includes the Cookie in the Message that creates the Hash.

Allows Challenger to apply additional check with Cookie; (and close connection if it does not pass).

Both ends have high probability MPA identification

Requires clear understanding of which end goes first; does not work if all of A-C sequences are allowed; does not work for Active/Active direct to iWARP peer startups.

Challenge Key (1st to transmit)

Key: 16 char text string “MPA framing mode”

Cookie: A 32 bit Randomly chosen number

M: Marker bit (1=No marker, 0=Marker)

C: CRC bit (1=No CRC desired, 0=CRC required)

Rev: MPA revision (0 for this version)

Reserved: transmit as zero, checked for zero

Challenge/Response Key (response)

RSA MD5 Hash of:

- Cookie concatenated with
- "MPA framing mode"

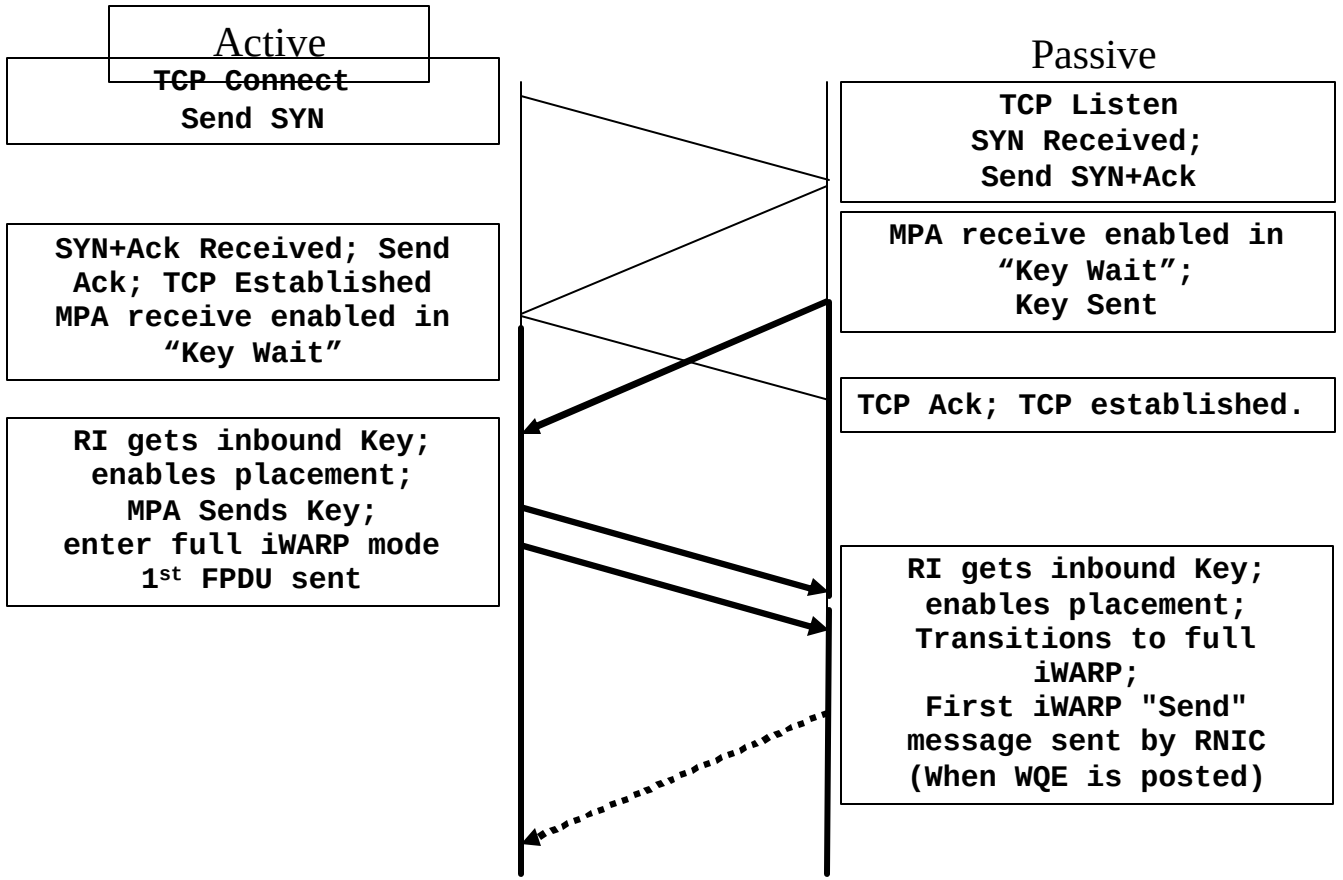
M: Marker bit (1=No marker, 0=Marker)

C: CRC bit (1=No CRC desired, 0=CRC required)

Rev: MPA revision (0 for this version)

Reserved: transmit as zero, checked for zero

IWarp Startup sequence (D)

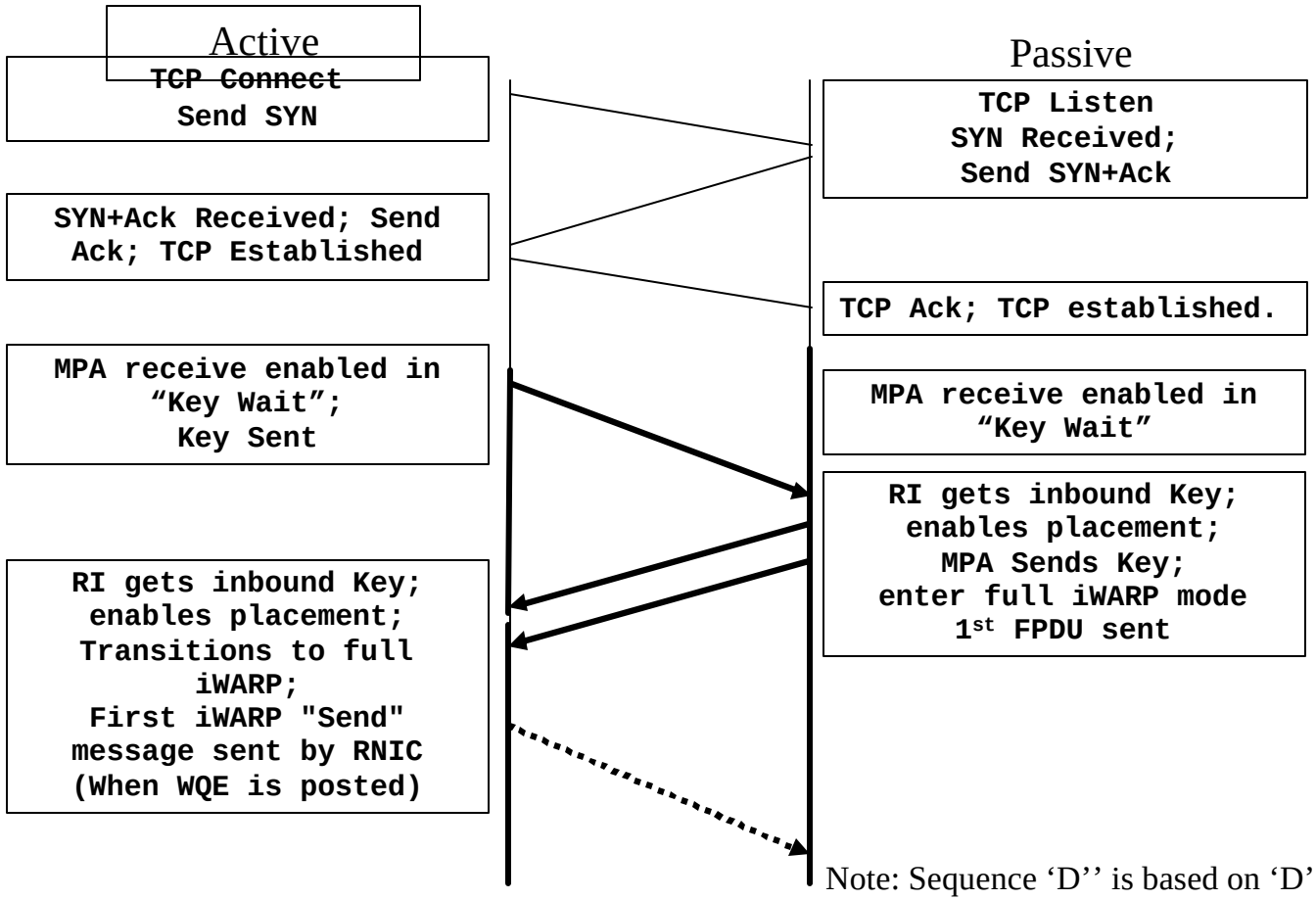


Note: Sequence 'D' is based on 'B'

D: using Challenge/Response

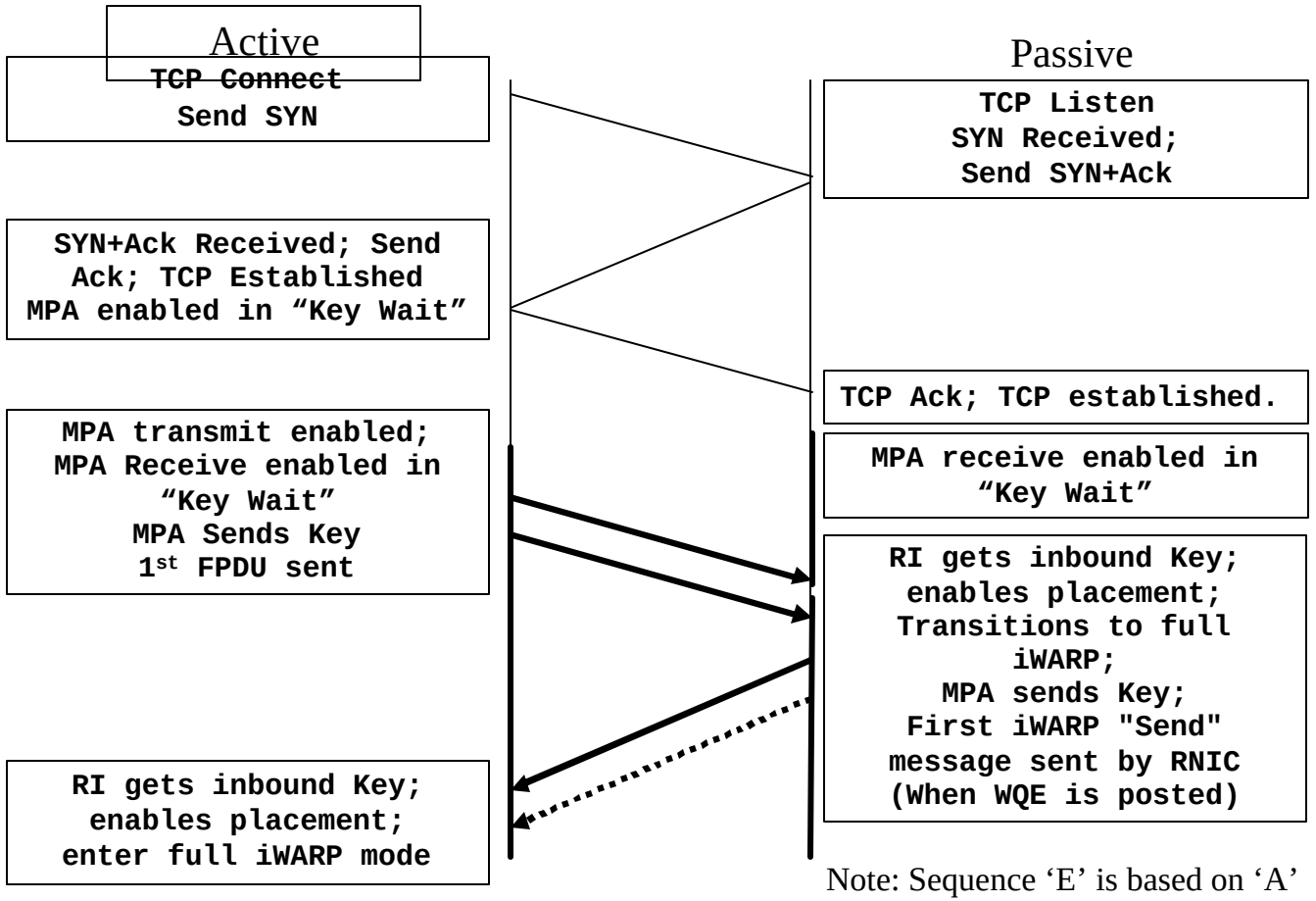
- An aggressive implementation of passive side (server?) can combine the TCP SYN+Ack and the MPA Challenge key
  - An aggressive implementation of the Active side (client?) can then combine the Response Key with the Ack to the SYN+Ack.
    - This results in a 4way exchange similar to SCTP
- Starting the full iWARP connection can be delayed until after Response Key arrives
  - Potential DOS remediation…
  - But requires new Verbs to send/receive keys prior to allocating and tying the QP.
    - Could use standard sockets, except that must be MPA's responsibility… (or does it?)

IWarp Startup sequence (D')



Note: Sequence 'D'' is based on 'D' but 1st to send is flipped.

IWarp Startup sequence (E)



Note: Sequence 'E' is based on 'A'