Summary of P802.1CQ/D0.5 in Task Group Ballot, July 2020

Roger Marks (EthAirNet Associates)
Antonio de la Oliva (InterDigital)
2020-07-15
P802.1CQ Administrative Status

- P802.1CQ Multicast and Local Address Assignment
  - [https://1.ieee802.org/tns/802-1cq/](https://1.ieee802.org/tns/802-1cq/)
- PAR approved 2016-02-05
  - renewed 2020-06-03; expires 2022-12-31
- assigned to TSN, 2019-07 (previously DCB, then OmniRAN)
- 802.1 WG Motion (approved 2019-07-18):
  - 802.1 authorizes Roger Marks, the Editor of P802.1CQ Multicast and Local Address Assignment to prepare drafts for and conduct Task Group balloting
- Draft 0.3 released 2019-10-21
- TSN last updated in presentation of 2020-03-19
- Draft 0.5 released 2020-06-30
  - 49 pages, plus frontmatter
  - Task Group Ballot, 2020-06-30 through 2020-07-31
Key PAR Content

• Scope
  – This standard specifies protocols, procedures, and management objects for locally-unique assignment of 48-bit and 64-bit addresses in IEEE 802 networks. Peer-to-peer address claiming and address server capabilities are specified.

• Need for the Project:
  – Currently, global addresses are assigned to most IEEE 802 end station and bridge ports. Increasing use of virtual machines and Internet of Things (IoT) devices could exhaust the global address space. To provide a usable alternative to global addresses for such devices, this project will define a set of protocols that will allow ports to automatically obtain a locally-unique address in a range from a portion of the local address space. Multicast flows also need addresses to identify the flows. They will benefit from a set of protocols to distribute multicast addresses. Peer-to-peer address claiming and address server capabilities will be included to serve the needs of smaller (e.g. home) and larger (e.g. industrial plants and building control) networks.
Key Points of P802.1CQ/D0.5

• Includes MAC Address Acquisition Protocol (MAAP) from IEEE Std 1722-2016, with minor error corrections, as one method
  – Claiming only (no client-server assignment)

• New Protocol for Assignment of Local and Multicast Addresses (PALMA)
  – does not use MAAP Ethertype or MAAP address ranges
  – both claiming and client-server assignment

• Messaging begins with DISCOVER (multicast from client; seeks server or peer)
  – client may adopt a source address chosen randomly from a specified range
    o note that we call it “client” even when there is no server
    o client may specify the requested range of addresses

• If server hears DISCOVER:
  – Server sends OFFER to client
  – Client sends REQUEST to server
  – Server sends assignment in ACK to client

• If peer hears DISCOVER claim for address already in use:
  – Peer sends DEFEND to client
  – Client sends multicast ANNOUNCE (stating self-assigned use of undefended claim)
### PALMA Client/Server Message Exchange

#### DISCOVERY phase,
for both claiming and server discovery

<table>
<thead>
<tr>
<th>Client 1</th>
<th>Server 1</th>
<th>Client 2</th>
<th>Server 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISCOVER [random1, mcast1] (setX)</td>
<td>random1 (\not\in) setX</td>
<td>DISCOVER [random2, mcast1] (setX)</td>
<td>random2 (\not\in) setX</td>
</tr>
<tr>
<td>DISCOVER [random1, mcast1] (setX)</td>
<td></td>
<td>DISCOVER [random2, mcast1] (setX)</td>
<td></td>
</tr>
<tr>
<td>OFFER [server1, random1] (setY1)</td>
<td></td>
<td>OFFER [server2, random2] (setY2)</td>
<td></td>
</tr>
<tr>
<td>REQUEST [unicast1, server1] (setY1)</td>
<td></td>
<td>REQUEST [unicast1, server1] (setY1)</td>
<td></td>
</tr>
<tr>
<td>ACK [server1, unicast1] (setY1)</td>
<td></td>
<td>ACK [server1, unicast1] (setY1)</td>
<td></td>
</tr>
</tbody>
</table>

Note: If client address is from the random set, then setY includes at least one unicast address, for the client address.
PALMA Client/Server Message Exchange

Client 1

<table>
<thead>
<tr>
<th>DISCOVER [random1, mcast1] (setX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISCOVER [random2, mcast1] (setX)</td>
</tr>
</tbody>
</table>

Server 1

<table>
<thead>
<tr>
<th>random1 ∉ setX</th>
</tr>
</thead>
<tbody>
<tr>
<td>random2 ∉ setX</td>
</tr>
</tbody>
</table>

Client 2

<table>
<thead>
<tr>
<th>OFFER [server1, random1] (setY1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFER [server2, random2] (setY2)</td>
</tr>
</tbody>
</table>

Server 2

Note: If client address is from the random set, then setY includes at least one unicast address, for the client address.

receiving OFFERs from multiple servers

REQUEST [unicast1, server1] (setY1)

ACK [server1, unicast1] (setY1)
PALMA Client/Server Message Exchange

DISCOVER [random1, mcast1] (setX)

random1 ∉ setX

DISCOVER [random2, mcast1] (setX)

random2 ∉ setX

OFFER [server1, random1] (setY1)

Note: If client address is from the random set, then setY includes at least one unicast address, for the client address.

OFFER [server2, random2] (setY2)

REQUEST [unicast1, server1] (setY1)

unicast1 ∈ setY1

ACK [server1, unicast1] (setY1)

REQUEST to to a selected server
DISCOVERY phase, for both claiming and server discovery

- DISCOVER [random1, mcast1] (setX)
  - random1 $\not\in$ setX
- DISCOVER [random2, mcast1] (setX)
  - random2 $\not\in$ setX
- DISCOVER [random3, mcast1] (setX)
  - random3 $\not\in$ setX
- DISCOVER [random4, mcast1] (setZ)
- DEFEND [client1, random4] (setZ)
- ANNOUNCE [client1, mcast1] (setY)
  - setY $\in$ setX
  - client1 $\in$ setY

setZ overlaps setY
PALMA Claiming Message Exchange

Client 3  Client 1  Client 2

DISCOVER [random1, mcast1] (setX)  random1 \notin setX

DISCOVER [random2, mcast1] (setX)  random2 \notin setX

DISCOVER [random3, mcast1] (setX)  random3 \notin setX

ANNOUNCE [client1, mcast1] (setY)  setY \in setX  client1 \in setY

DISCOVER [random4, mcast1] (setZ)  setZ overlaps setY

ANNOUNCE [client1, mcast1] (setY)

DEFEND [client1, random4] (setZ)
PALMA Claiming Message Exchange

Client 3

Client 1

DISCOVER [random1, mcast1] (setX)  random1 ∉ setX

DISCOVER [random2, mcast1] (setX)  random2 ∉ setX

DISCOVER [random3, mcast1] (setX)  random3 ∉ setX

ANNOUNCE [client1, mcast1] (setY)  setY ∈ setX

client1 ∈ setY

Client 2

DISCOVER [random4, mcast1] (setZ)

DEFEND [client1, random4] (setZ)

setZ overlaps setY

DEFEND the self-assignment
Address Pools

- Each server needs a disjoint address pool
- Address pool for claiming is disjoint from all server pools
- Device without an address chooses from range beginning with 2A-00.
- A device may issue a DISCOVER proposing a prior PALMA-assigned address.
- If a server offers an address, device is prohibited from self-assigning.

### Table 1—PALMA local address assignment ranges

<table>
<thead>
<tr>
<th>Usage</th>
<th>Initial Address Octets</th>
<th>SLAP Quadrant</th>
<th>PALMA assignment method</th>
<th>Address count limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>unicast</td>
<td>0A</td>
<td>11 (SAI)</td>
<td>self-assignment</td>
<td>16</td>
</tr>
<tr>
<td>unicast</td>
<td>1A</td>
<td>11 (SAI)</td>
<td>server assignment</td>
<td>-</td>
</tr>
<tr>
<td>multicast</td>
<td>0B</td>
<td>11 (SAI)</td>
<td>self-assignment</td>
<td>-</td>
</tr>
<tr>
<td>multicast</td>
<td>1B</td>
<td>11 (SAI)</td>
<td>server assignment</td>
<td>-</td>
</tr>
<tr>
<td>unicast or multicast</td>
<td>per assigned CID</td>
<td>01 (ELI)</td>
<td>server assignment, in accordance with authorization per assigned CID</td>
<td>-</td>
</tr>
<tr>
<td>unicast or multicast</td>
<td>any</td>
<td>00 (AAI)</td>
<td>server assignment</td>
<td>-</td>
</tr>
</tbody>
</table>
Bypass DISCOVERY when a server has been previously identified and stored.

Prior server assignment can be also be recalled from cold-storage for quick renewal or reactivation.
PALMA Client
Flow Diagram: DISCOVERY State
PALMA Client Flow Diagram: REQUESTING State

PALMA Client

Flow Diagram:

REQUESTING State

- Enter Client REQUESTING state from Client DISCOVERY, DEFENDING, or BOUND State
- Stop listening to PALMA multicast address
- Select request set
- Initialize Request Repeat Count
- Decrement Request Repeat Count
- If Repeat Count ≠ 0
  - Issue REQUEST
  - Start Request Timer
  - Receive ACK
    - If Yes
      - Yes: Error?
        - If No
          - Enter Client BOUND state
        - If Yes: Request Timer Timeout
    - If No
      - Enter Client REQUESTING state from Client DISCOVERY, DEFENDING, or BOUND State
    - No: Request Timer Timeout
  - If No
    - Enter Client BOUND state
PALMA Client
Flow Diagram: BOUND State

- Enter Client BOUND state from Client REQUESTING State
- Start Assignment Timer
- Assignment Timer Timeout
  - No
  - Yes
- START/ Restart
PALMA Client Flow Diagram: DEFENDING State
# PALMA Client State Machine

<table>
<thead>
<tr>
<th>Event</th>
<th>Start</th>
<th>Discovery</th>
<th>Requesting</th>
<th>Bound</th>
<th>Defending</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restart!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Begin!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Known server!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Disc Add!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Offer!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Disc Timer!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Disc Count!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ack!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ack Time!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Disc Timer!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Port Op!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Release!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Table 27—PALMA Client State Machine

<table>
<thead>
<tr>
<th>Event</th>
<th>Start</th>
<th>Discovery</th>
<th>Requesting</th>
<th>Bound</th>
<th>Defending</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Disc Add!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Disc Timer!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Disc Count!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ack!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ack Time!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Disc Timer!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Port Op!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Release!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Next Steps

• Collect and address Task Group Ballot feedback
  – Please review and comment. All comments welcome.

• Recheck and refine logic
  – Carefully specify which fields are stored and recalled at each action
  – Working on a reference implementation of D0.5 to identify bugs

• Document specific processes for wireless LAN case

• Create a table or figure to show the relationship among address sets

• Complete some missing background sections

• Explain the IETF DHCPv6 draft on assigning MAC addresses