INTAREA
Internet-Draft
Intended status: Informational
Expires: December 2, 2019

S. Kanugovi
Nokia
F. Baboescu
Broadcom
J. Zhu
Intel
J. Mueller
AT&T
S. Seo
Korea Telecom
May 31, 2019

Multiple Access Management Services
draft-kanugovi-intarea-mams-framework-04

Abstract

In multiconnectivity scenarios, the end-user devices can simultaneously connect to multiple networks based on different access technologies and network architectures like WiFi, LTE, DSL. Both the quality of experience of the users and the overall network utilization and efficiency may be improved through the smart selection and combination of access and core network paths that can dynamically adapt to changing network conditions. This document presents a unified problem statement and introduces a solution for managing multiconnectivity. The solution has been developed by the authors based on their experiences in multiple standards bodies including the IETF and 3GPP, but is not an Internet Standard and does not represent the consensus opinion of the IETF. This document describes the requirements, solution principles, and an architectural framework that aims to provide best performance while being easy to implement in a wide variety of multiconnectivity deployments. It specifies the protocol multi-access management to: 1) flexibly select the best combination of access and core network paths for uplink and downlink; as well as 2) determine the user plane treatment and traffic distribution over the selected links ensuring network efficiency and application performance.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 2, 2019.

Table of Contents

1. Introduction

Multi Access Management Services (MAMS) is a programmable framework that provides mechanisms for flexible selection of network paths in a multi-access communication environment, based on application needs. It leverages network intelligence and policies to dynamically adapt traffic distribution across selected paths and user plane treatment to changing network/link conditions. The network path selection and configuration messages are carried as user plane data between the functional elements in the network and the end-user device, and thus without any impact to the control plane signaling schemes of the underlying access network(s). For example, in a multi-access network with LTE and WiFi technologies, existing LTE and existing WiFi signaling procedures will be used to setup the LTE and WiFi connections, respectively, and MAMS specific control plane messages are carried as LTE or WiFi user plane data. The MAMS framework defined in this document provides the capabilities of smart selection and flexible combination of access paths and core network paths, as well as the user plane treatment when the traffic is distributed across the selected paths. Thus, it is a broad programmable framework providing functions beyond simple sharing of network policies such as provided by Access Network Discovery and Selection Function (ANDSF) [ANDSF] that offers policies and rules for assisting 3GPP devices to discover and select available access networks. Further, it allows the choice and configuration of user plane treatment for the traffic over the multiple paths, depending on the needs of the application.

MAMS mechanisms are not dependent on any specific access network type or user plane protocols like TCP, UDP, GRE, MPTCP etc. It co-exists and complements the existing protocols by providing a way to negotiate and configure these protocols based on client and network capabilities per access basis to match their use for a given multi-access scenario. Further, it allows load balancing of the traffic flows across the selected multiple accesses and exchange of network state information to be used for network intelligence to optimize the performance of such protocols.

The document presents the requirements, solution principles, functional architecture, and protocols for realizing the MAMS framework. An important goal for MAMS is to ensure that it either requires minimum dependency or (better) no dependency on the actual access technologies of the participating links, beyond the fact that MAMS functional elements form an IP-overlay across the multiple paths. This allows the scheme to be future proof by allowing independent technology evolution of the existing access and core networks as well as, seamless integration of new access technologies.

The solution described in this document has been developed by the authors based on their experiences in multiple standards bodies including the IETF and 3GPP, but is not an Internet Standard and does not represent the consensus opinion of the IETF.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

"Client": The end-user device supporting connections with multiple access nodes, possibly over different access technologies.

"Multiconnectivity Client": A client with multiple network connections.

"Access network": The segment in the network that delivers user data packets to the client via an access link like WiFi airlink, LTE airlink, or DSL.

"Core": The functional element that anchors the client IP address used for communication with applications via the network.

"Network Connection manager"(NCM): A functional entity in the network that handles MAMS control messages from the client and configures distribution of data packets over the multiple available access and core network paths, and user plane treatment of the traffic flows.

"Client Connection Manager" (CCM): A functional entity in the client that exchanges MAMS Signaling with the Network Connection Manager and configures the multiple network paths at the client for transport of user data.

"Network Multi Access Data Proxy" (N-MADP): This functional entity in the network handles the user data traffic forwarding across multiple network paths. N-MADP is responsible for MAMS related user-plane functionalities in the network.

"Client Multi Access Data Proxy" (C-MADP): This functional entity in the client handles the user data traffic forwarding across multiple network paths. C-MADP is responsible for MAMS related user-plane functionalities in the client.

"Anchor Connection": Refers to the network path from the N-MADP to the user plane gateway (IP anchor) that has assigned an IP address to the client.

"Delivery Connection": Refers to the network path from the N-MADP to the client.

3. Problem Statement

Typically, an end-user device has access to multiple communication networks based on different technologies, say LTE, WiFi, DSL, MuLTEfire, for accessing application services. Different technologies exhibit benefits and limitations in different scenarios. For example, WiFi provides high throughput for end users when under good coverage, but the throughput degrades significantly as the user moves closer to the edge of WiFi coverage (typically in the range of few tens of meters) or with large user population (due to contention based WiFi access scheme). In LTE networks, the capacity is often constrained by the limited availability of licensed spectrum. However, the quality of the service is predictable even in multi-user scenarios due to controlled scheduling and licensed spectrum usage.

Additionally, the use of a particular access network path is often coupled with the use of its associated core network and the services that are offered by it. For example, in an enterprise that has deployed both WiFi and LTE networks, the enterprise services, like printers, Corporate Audio and Video conferencing, are accessible only via WiFi access connected to the enterprise hosted (WiFi) core, whereas the LTE access can be used to get operator core anchored services including access to public Internet.

Thus, application performance in different scenarios becomes dependent on the choice of the access networks (e.g. WiFi, LTE, etc.) and the used network and transport protocols (e.g. VPN, MPTCP, GRE etc.). Therefore, to achieve the best possible application performance in a wide range of scenarios, a framework is needed that allows the selection and flexible combination of access and core network paths and used protocols for uplink and downlink data delivery.

For example, to ensure best performance for enterprise applications at all times, in uncongested scenarios, when the user is under good WiFi coverage, it would be beneficial to use WiFi access in both uplink and downlink for connecting to enterprise applications. However, in congested scenarios or when the user is getting close to the edge of its WiFi coverage, the use of WiFi in uplink by multiple users can lead to degraded capacity and increased delays due to contention. In this case, it would be beneficial to at least use the

LTE access for increased uplink coverage while WiFi may still continue to be used for downlink.

4. Requirements

The requirements set out in this section are for the definition of behavior of the MAMS mechanism and the related functional elements.

4.1. Access Technology Agnostic Interworking

The access nodes may use different technology types like LTE, WiFi, etc. The framework, however, MUST be agnostic to the type of underlying technology used at the access network.

4.2. Support Common Transport Deployments

The network path selection and user data distribution MUST work transparently across various transport deployments that include end-to-end IPsec, VPNs, and middleboxes like NATs and proxies.

4.3. Independent Access Path Selection for Uplink and Downlink

A Client SHOULD be able to transmit on the uplink and, receive on the downlink, using one or more accesses. The selection of the access paths for uplink and downlink SHOULD happen independent of each other.

4.4. Core Selection Independent of Uplink and Downlink Access

A client SHOULD flexibly select the Core, independent of the access paths used to reach the Core, depending on the application needs, local policies and the result of MAMS control plane negotiation.

4.5. Adaptive Access Network Path Selection

The framework MUST have the ability to determine the quality of each of the network paths, e.g. access link delay and capacity. The network path quality information needs to be considered in the logic for selection of the combination of network paths to be used for transporting user data. The path selection algorithm can use network path quality information, in addition to other considerations like network policies, for optimizing network usage and enhancing QoE delivered to the user.

4.6. Multipath Support and Aggregation of Access Link Capacities

The framework MUST support distribution and aggregation of user data across multiple network paths at the IP layer. The client SHOULD be able to leverage the combined capacity of the multiple network connections by enabling simultaneous transport of user data over multiple network paths. If required, packet re-ordering needs to be done at the receiver. The framework MUST allow flexibility to choose the flow steering and aggregation protocols based on capabilities supported by the client and the network data plane entities. The multi-connection aggregation solution MUST support existing transport and network layer protocols like TCP, UDP, GRE. The framework MUST allow use and configuration of existing aggregation protocols such as Multi-Path TCP(MPTCP) and SCTP.

4.7. Scalable Mechanism based on User Plane Interworking

The framework MUST leverage commonly available transport, routing and tunneling capabilities to provide user plane interworking functionality. The addition of functional elements in the user plane path between the client and the network MUST not impact the access technology specific procedures. This makes solution easy to deploy and scale when different networks are added and removed.

4.8. Separate Control and Data Plane functions

The client MUST use the control plane protocol to negotiate with the network, the choice of access and core network paths for both uplink and downlink, as well as the user plane protocol treatment. The control plane MUST configure the actual user plane data distribution function per this negotiation. A common control protocol SHOULD allow creation of multiple user plane function instance with potentially different user plane (e.g. tunneling) protocol types. This enables maintaining a clear separation between the control and data plane functions, allowing the framework to be scalable and extensible, e.g. using SDN based architecture and implementations.

4.9. Lossless Path (Connection) Switching

When switching data traffic from one path (connection) to another, packets may be lost or delivered out-of-order, which will have negative impacts on the performance of higher layer protocols, e.g. TCP. The framework SHOULD provide necessary mechanisms to ensure in-order delivery at the receiver, e.g. during path switching. The framework MUST not cause any packet loss beyond that of access network mobility functions may cause.

4.10. Concatenation and Fragmentation for adaptation to MTU Differences

Different network paths may have different security and middlebox (e.g NAT) configurations, which will lead to use of different tunneling protocols for transport of data between the network user plane function and the client. As a result, different effective payload sizes (e.g. due to variable encapsulation header overheads) per network path are possible. Hence, MAMS framework SHOULD support fragmentation of a single IP packet payload across MTU sized IP packets to avoid IP fragmentation when aggregating packets from different paths. Further, concatenation of multiple IP packets into a single IP packet to improve efficiency in packing the MTU size should also be supported.

4.11. Configuring Network Middleboxes based on Negotiated Protocols

The framework SHOULD enable identification of the optimal parameters that may be used for configuring the middle-boxes, like radio link dormancy timers, binding expiry times and supported MTUs, for efficient operation of the user plane protocols, based on parameters negotiated between the client and the network, e.g. Configuring longer binding expiry time in NATs when UDP transport is used in contrast to the scenario where TCP is configured at the transport layer.

4.12. Policy based Optimal Path Selection

The framework MUST support consideration of policies at the client, in addition to guidance from the network, for network path selection addressing different application requirements.

4.13. Access Technology Agnostic Control Signaling

The control plane signaling MUST NOT be dependent on the underlying access technology procedures, e.g. be carried transparently as user plane. It should support delivery of control plane signaling over the existing Internet protocols, e.g. TCP or UDP.

4.14. Service Discovery and Reachability

There can be multiple instances of the control and user plane functional elements of the framework, either collocated or hosted on separate network elements, and reachable via any of the available user plane paths. The client MUST have flexibility to choose the appropriate control plane instance in the network and use the control plane signaling to choose the desired user plane functional element instances. The choice can be based on considerations like, but not

limited to, quality of link through which the network function is reachable, client preferences, pre-configuration etc.

5. Solution Principles

This document proposes the Multiple Access Management Services(MAMS) framework for dynamic selection and flexible combination of access and core network paths independently for the uplink and downlink, as well as the user plane treatment for the traffic spread across the selected links. MAMS framework consists of clearly separated control and user plane functions in the network and the client. The control plane protocol allows configuration of the user plane protocols and desired network paths for transport of application traffic. The control plane messages are carried as user plane data over any of the available network paths between the peer control plane functional elements in the client and the network . Multiple user plane paths are dynamically distributed across multiple access networks and aggregated in side the common core network. The access network diversity is not exposed to the application servers but kept within the scope of the elements defined in this framework. This offloads the application servers from reacting to access link changes caused to mobility events or changing of link characteristics. The selection of paths and user plane treatment of the traffic, is based on negotiation of capabilities (of device and network) and probing of network link quality between the user plane functional elements at the end-user device/client and the network. The framework enables leveraging network intelligence to setup and dynamically configure the best access network path combination based on device and network capabilities, application needs and knowledge of the network state.

6. MAMS Reference Architecture

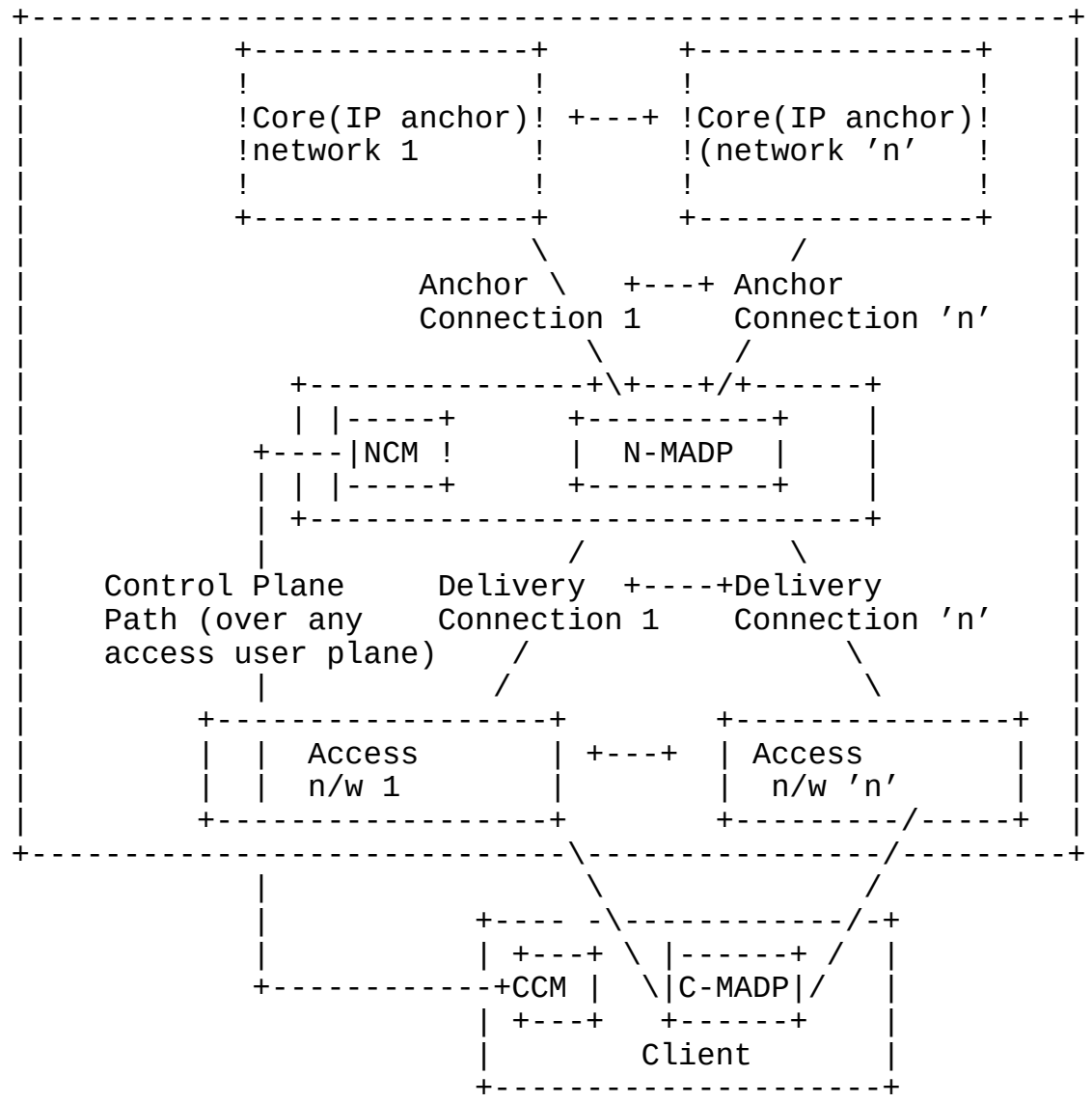

Figure 1: MAMS Reference Architecture

Figure 1 illustrates MAMS architecture for the scenario of a client served by multiple (n) networks. It introduces the following functional elements,

- o Network Connection Manager (NCM) and Client Connection Manager (CCM) in the control plane, and
- o Network Multi Access Data Proxy (N-MADP) and Client Multi Access Data Proxy (C-MADP) handling the user plane.

NCM: It is the functional element in the network that handles the MAMS control plane procedures. It configures the network (N-MADP)

and client (C-MADP) user plane functions like negotiating the client on the use of available access network paths, protocols and rules for processing the user plane traffic, as well as link monitoring procedures. The control plane messages between the NCM and CCM are transported as an overlay, without any impact to the underlying access networks.

CCM: It is the peer functional element in the client for handling MAMS control plane procedures. It manages multiple network connections at the client. It is responsible for exchange of MAMS signaling messages with the NCM for supporting functions like UL and DL user network path configuration for transporting user data packets, link probing and reporting to support adaptive network path selection by NCM. In the downlink, for the user data received by the client, it configures C-MADP such that application data packet received over any of the accesses to reach the appropriate application on the client. In the uplink, for the data transmitted by the client, it configures the C-MADP to determine the best access links to be used for uplink data based on a combination of local policy and network policy delivered by the NCM.

N-MADP: It is the functional element in the network that handles the user data traffic forwarding across multiple network paths, as well as other user-plane functionalities like encapsulation, fragmentation, concatenation, reordering, retransmission, etc. It is the distribution node that routes the uplink user plane traffic to the appropriate anchor connection towards the core network, and the downlink user traffic to the client over the appropriate delivery connection(s). In the downlink, the NCM configures the use of delivery connections, and user plane protocols at the N-MADP for transporting user data traffic. The N-MADP should implement ECMP support for the down link traffic. Or alternatively, it may be connected to a router with ECMP functionality. The load balancing algorithm at the N-MADP is configured by the NCM, based on static and/or dynamic network policies like assigning access and core paths for specific user data traffic type, data volume based percentage distribution, and link availability and feedback information from exchange of MAMS signaling with the CCM at the Client.. N-MADP can be configured with appropriate user plane protocols to support both per-flow and per-packet traffic distribution across the delivery connections. In the uplink, N-MADP selects the appropriate anchor connection over which to forward the user data traffic, received from the client (via the delivery connections). The forwarding rules in the uplink at the N-MADP are configured by the NCM based on application requirements, e.g. Enterprise hosted Application flows via Wi-Fi Anchor, Mobile Operator hosted applications via the Cellular Core.

C-MADP: It is the functional element in the client that handles the MAMS user plane data procedures. C-MADP is configured by CCM based on signaling exchange with NCM and local policies at the client. The CCM configures the selection of delivery connections and the user plane protocols to be used for uplink user data traffic based on the signaling exchanged with NCM. The C-MADP entity handles user plane data forwarding across multiple delivery connections and associated user-plane functions like encapsulation, fragmentation, concatenation, reordering, retransmissions, etc.

The NCM and N-MADP can be either collocated or instantiated on different network nodes. NCM can setup multiple N-MADP instances in the network. NCM controls the selection of N-MADP instance by the client and the rules for distribution of user traffic across the N-MADP instances., This is beneficial in multple deployment scenarios, like the following examples.

- Different N-MADP instances to handle different sets of clients for load balancing across clients
- Address deployment topologies e.g. N-MADP hosted at the user plane node at the access edge or in the core network, while the NCM hosted at the access edge node)
- Address access network technology architecture. For exanple, N-MADP instance at core network node to manage traffic distribution across LTE and DSL networks, and N-MADP instance at access network node to manage traffic distribution across LTE and Wi-Fi traffic.
- A single client can be configured to use multiple N-MADP instances. This is beneficial in addressing different application requirements. For example, separate N-MADP instances to handle TCP and UDP transport based traffic.

Thus, MAMS architecture flexibly addresses multiple network deployments.

7. MAMS Protocol Architecture

This section describes the protocol structure for the MAMS User and Control plane functional elements.

7.1. MAMS Control-Plane Protocol

Figure 2 shows the default MAMS control plane protocol stack. WebSocket is used for transporting management and control messages between NCM and CCM.

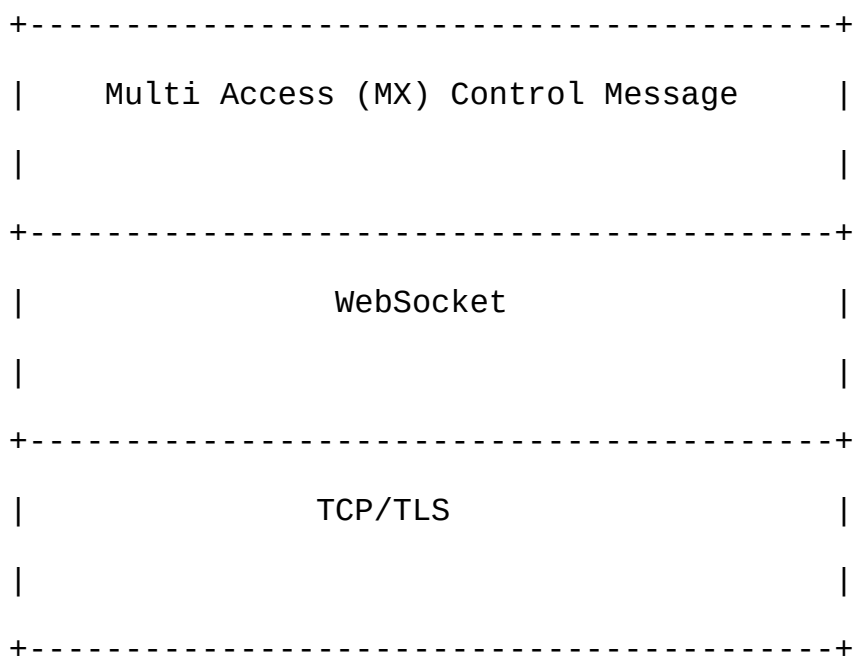

Figure 2: TCP-based MAMS Control Plane Protocol Stack

7.2. MAMS User Plane Protocol

Figure 3 shows the MAMS user plane protocol stack.

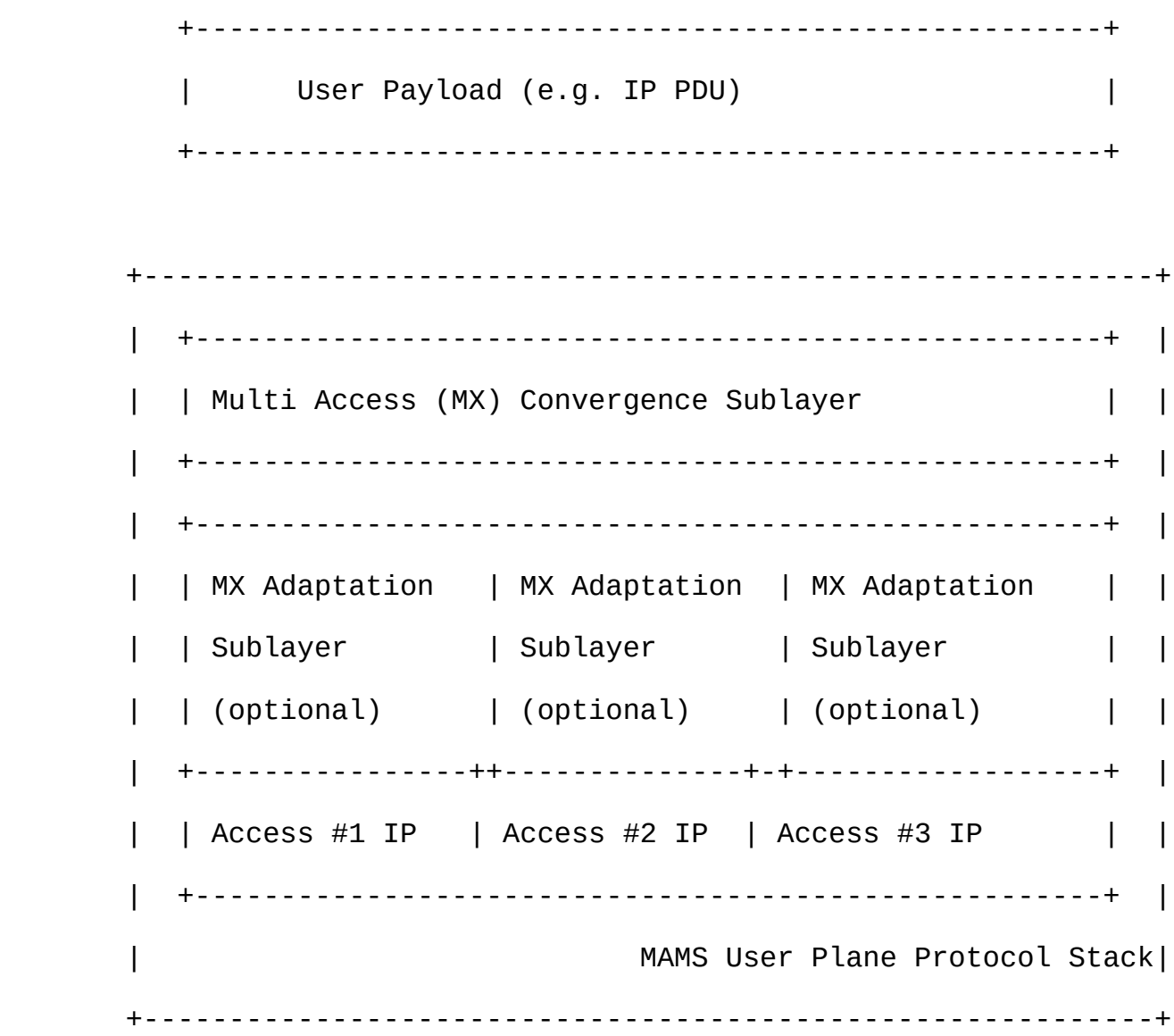

Figure 3: MAMS User Plane Protocol Stack

It consists of the following two Sublayers:

o Multi-Access (MX) Convergence Sublayer: The MAMS framework configures the Convergence sublayer to perform multi-access specific tasks in the user plane. This layer performs functions

like access (path) selection, multi-link (path) aggregation, splitting/reordering, lossless switching, fragmentation, concatenation, etc. MX Convergence layer can be implemented using existing user plane protocols like Multipath TCP (MPTCP [RFC 6824]), Multi Path QUIC (MPQUIC [I-D.deconinck-multipath-quic]) or by adapting encapsulating header/trailer schemes like Generic Routing and Encapsulation (GRE [RFC 2784], [RFC 2890]), Generic Multi Access(GMA [I-D.zhu-intarea-gma]).

- o Multi-Access (MX) Adaptation Sublayer: The MAMS framework configures the Adaptation Sublayer to address transport network related aspects like reachability and security in the user plane. This layer performs functions to handle tunnelling, network layer security, and NAT. MX Adaptation can be implemented using IPsec, DTLS or Client NAT (Source NAT at Client with inverse mapping at N-MADP [I-D.zhu-intarea-mams-user-protocol]). The MX Adaptation Layer is optional and can be independently configured for each of the Access Links. E.g. In a deployment with LTE (assumed secure) and Wi-Fi (assumed not secure), the MX Adaptation Sublayer can be omitted for the LTE link but MX Adaptation Sublayer is configured as IPsec for securing the Wi-Fi link. Further details on the MAMS user plane are described in [I-D.zhu-intarea-mams-user-protocol].

8. MAMS Control Plane Procedures

8.1. Overview

CCM and NCM exchange signaling messages to configure the user plane functions, C-MADP and N-MADP, at the client and network respectively. The means for CCM to obtain the NCM credentials (FQDN or IP Address) for sending the initial discovery messages are out of the scope of MAMS document. As an example, the client can obtain the NCM credentials using methods like provisioning, DNS query. Once the discovery process is successful, the (initial) NCM can update and assign additional NCM addresses, e.g. based on MCC/MNC tuple information received in the MX Discovery Message, for sending subsequent control plane messages.

CCM discovers and exchanges capabilities with the NCM. NCM provides the credentials of the N-MADP end-point and negotiates the parameters for user plane with the CCM. CCM configures C-MADP to setup the user plane path (e.g. MPTCP/UDP Proxy Connection) with the N-MADP based on the credentials (e.g. (MPTCP/UDP) Proxy IP address and port, Associated Core Network Path), and the parameters exchanged with the NCM. Further, NCM and CCM exchange link status information to adapt traffic steering and user plane treatment with dynamic network conditions. The key procedures are described in details in the following sub-sections.

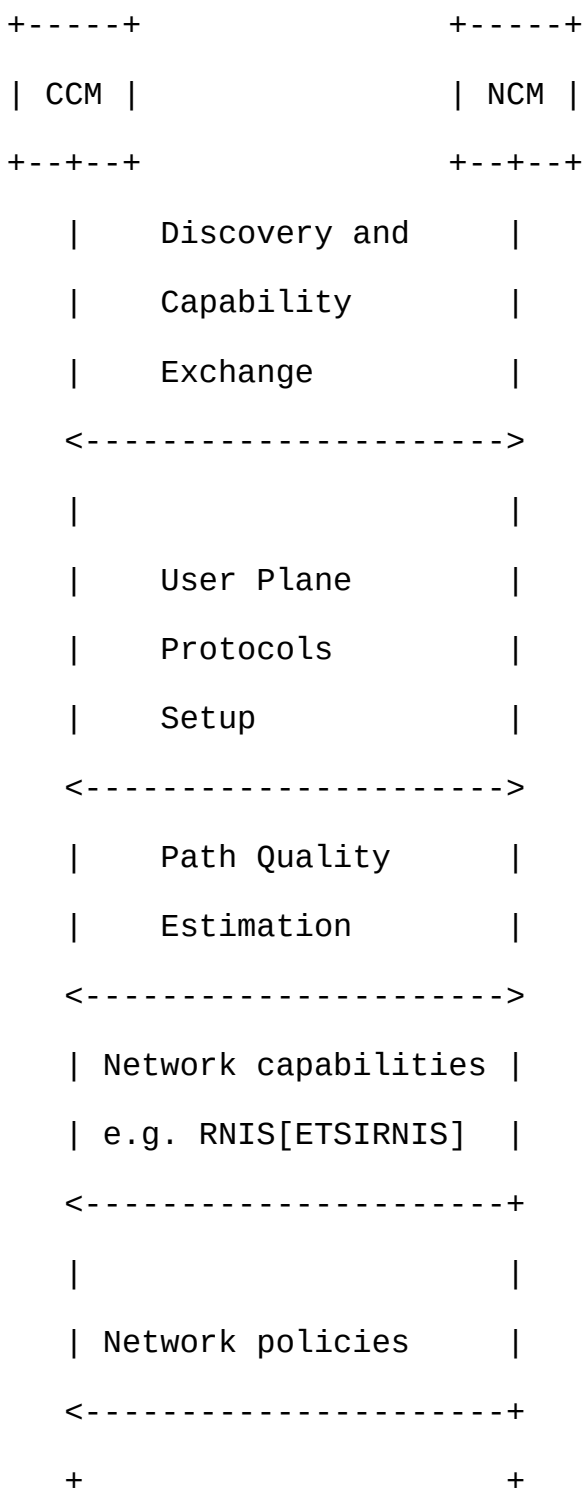
CCM
NCM
Discovery and
Capability
Exchange
User Plane
Protocols
Setup
Path Quality
Estimation
Network capabilities
e.g. RNIS[ETSIRNIS]
Network policies

Figure 4: MAMS Control Plane Procedures

8.2. Common fields in MAMS Control Messages

Each MAMS control message consists of the following common fields:

- Version: indicates the version of MAMS control protocol.
- Message Type: indicates the type of the message, e.g. MX Discovery, MX Capability REQ/RSP etc.
- Sequence Number: auto-incremented integer to uniquely identify a transaction of message exchange, e.g. MX Capability REQ/RSP.

8.3. Common Procedures for MAMS Control Messages

This section describes the common procedures for MAMS Control Messages.

8.3.1. Message Timeout

MAMS Control plane peer (NCM or CCM) waits for a duration of MAMS_TIMEOUT ms, after sending a MAMS control message, before timing out when expecting a response. The sender of the message will retransmit the message for MAMS_RETRY times before declaring failure. A failure implies that the MAMS peer is dead, and the sender reverts back to native non-multi access/single path mode. CCM may initiate the MAMS discovery procedure for re-establishment of the MAMS session.

8.3.2. Keep Alive Procedure

MAMS Control plane peers execute the keep alive procedures to ensure that peers are reachable and to recover from dead-peer scenarios. Each MAMS control plane end-point maintains a MAMS_KEEP_ALIVE timer that is set for duration MAMS_KEEP_ALIVE_TIMEOUT. MAMS_KEEP_ALIVE timer is reset whenever the peer receives a MAMS Control message. When MAMS_KEEP_ALIVE timer expires, MAMS KEEP ALIVE REQ message is sent. On reception of a MAMS KEEP ALIVE REQ message, the receiver responds with a MAMS KEEP ALIVE RSP message. If the sender does not receive a MAMS Control message in response to MAMS_RETRY number of retries of MAMS KEEP ALIVE REQ message, the MAMS peer declares that the peer is dead. CCM may initiate MAMS Discovery procedure for re-establishment of the MAMS session.

CCM shall additionally send MX KEEP ALIVE REQ message immediately to NCM whenever it detects a handover from one base station/access point to another. During this time the user equipment shall stop using MAMS user plane functionality in uplink direction till it receives a MX KEEP ALIVE RSP from NCM.

MX KEEP ALIVE REQ includes following information:

- Reason: Can be ’Timeout’ or ’Handover’. Reason ’Handover’ shall be used by CCM only on detection of handover.
- Unique Session Identifier: As defined in Section 8.4.
- Connection Id: This field shall be mandatorily be included if the reason is ’Handover’.
- Delivery Node Identity (ECGI in case of LTE and WiFi AP Id or MAC address in case of WiFi). This field shall be mandatorily be included if the reason is ’Handover’.

8.4. Discovery & Capability Exchange

Figure 5 shows the MAMS discovery and capability exchange procedure consisting of the following key steps:

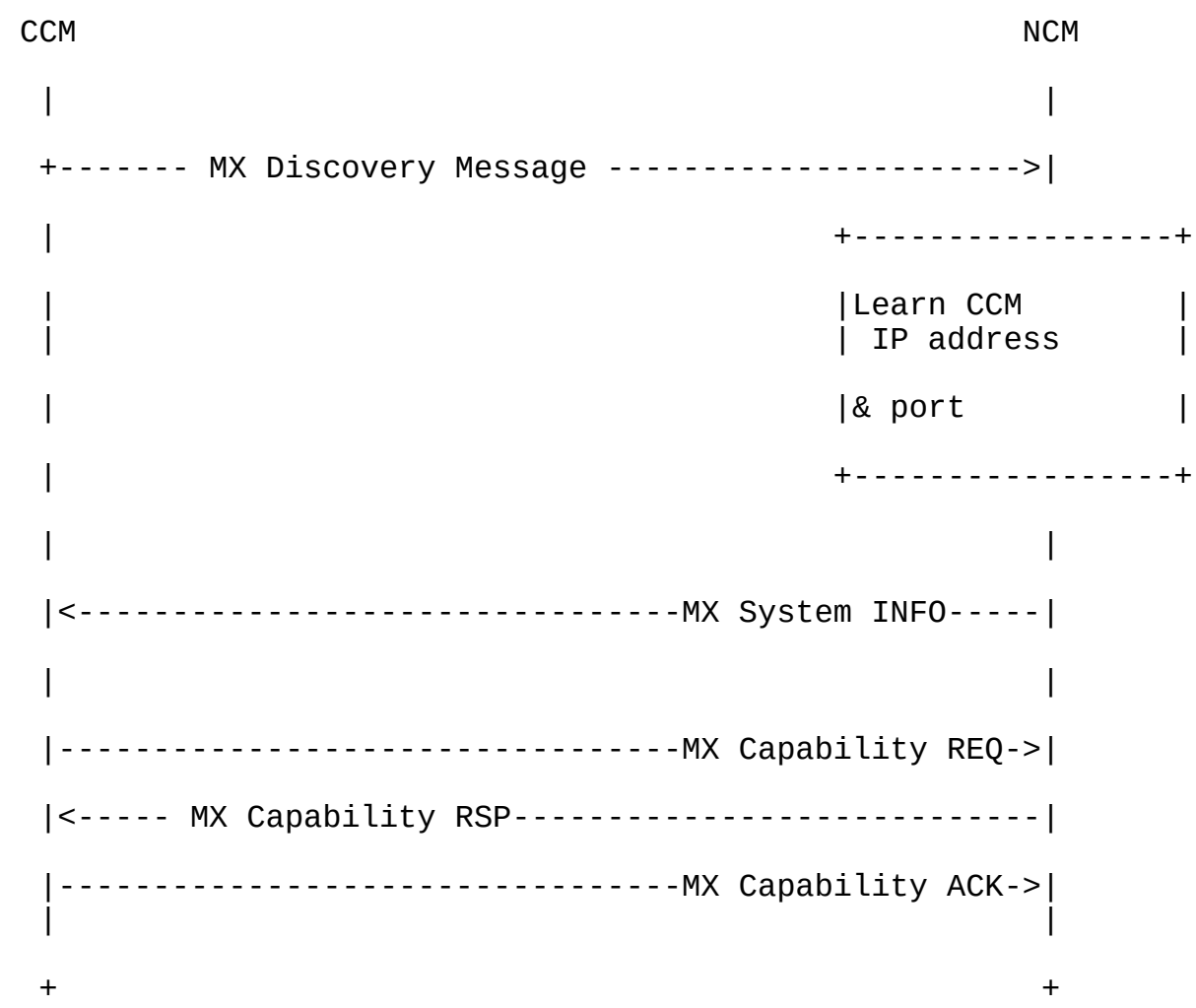

Figure 5: MAMS Control Procedure for Discovery & Capability Exchange

Step 1 (Discovery): CCM periodically sends out the MX Discovery Message to a pre-defined (NCM) IP Address/port until MX System INFO message is received in acknowledgement.

MX Discovery Message includes the following information:

- MAMS Version
- MCC/MNC Tuple: Optional Parameter to Identify the Operator Network to which the client is susbcribed, in conformance with format specified in [E212]

MX System INFO includes the following information:

- Number of Anchor Connections

  For each Anchor Connection, it includes the following parameters:

  * Connection ID: Unique identifier for the Anchor Connection
  * Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
  * NCM Endpoint Address (For Control Plane Messages over this connection)

    + IP Address or FQDN (Fully Qualified Domain Name)
    + Port Number

Step 2 (Capability Exchange): On receiving MX System Info message CCM learns the IP Address and port to start the step 2 of the control plane connection, and sends out the MX Capability REQ message, including the following Parameters:

- MX Feature Activation List: Indicates if the corresponding feature is supported or not, e.g. lossless switching, fragmentation, concatenation, Uplink aggregation, Downlink aggregation, Measurement, probing, etc.
- Number of Anchor Connections (Core Networks)

  For each Anchor Connection, it includes the following parameters:

  * Connection ID
  * Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
- Number of Delivery Connections (Access Links)

  For each Delivery Connection, it includes the following parameters:

  * Connection ID
  * Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
- MX Convergence Method Support List

  * GMA
  * MPTCP Proxy
  * GRE Aggregation Proxy
  * MPQUIC
- MX Adaptation Method Support List

  * UDP Tunnel without DTLS
  * UDP Tunnel with DTLS
  * IPsec Tunnel [RFC3948]

    * Client NAT

In response, NCM creates a unique identity for the CCM session, and sends out the MX Capability RSP message, including the following information:

- MX Feature Activation List: Indicates if the corresponding feature is enabled or not, e.g. lossless switching, fragmentation, concatenation, Uplink aggregation, Downlink aggregation, Measurement, probing, etc.
- Number of Anchor Connections (Core Networks)

  For each Anchor Connection, it includes the following parameters:

  * Connection ID
  * Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
- Number of Delivery Connections (Access Links)

  For each Delivery Connection, it includes the following parameters:

  * Connection ID
  * Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: Multi-Fire; 3: LTE)
- MX Convergence Method Support List

  * GMA
  * MPTCP Proxy
  * GRE Aggregation Proxy
  * MPQUIC
- MX Adaptation Method Support List

  * UDP Tunnel without DTLS
  * UDP Tunnel with DTLS
  * IPsec Tunnel [RFC3948]
  * Client NAT

Unique Session Identifier: Unique session identifier for the CCM which has setup the connection. In case the session for the UE already exists then the existing unique session identifier is sent back.

- NCM Id: Unique Identity of the NCM in the operator network.
- Session Id: Unique identity assigned to the CCM instance by this NCM instance.

In response to MX Capability RSP message, the CCM sends confirmation (or reject) in the MX Capability ACK message. MX Capability ACK includes the following parameters

- Unique Session Identifier: Same identifier as provided in MX Capability RSP.
- Acknowledgement: An indication if the client has accepted or rejected the capability phase.
  - MX ACCEPT: CCM Accepts the Capability set proposed by the NCM.
  - MX REJECT: CCM Rejects the Capability set proposed by the NCM.

If MX_REJECT is received by the NCM, the current MAMS session will be terminated.

If CCM can no longer continue with the current capabilities, it should send an MX SESSION TERMINATE message to terminate the MAMS session. In response, the NCM should send a MX SESSION TERMINATE ACK to confirm the termination.

8.5. User Plane Configuration

Figure 6 shows the user plane configuration procedure consisting of the following key steps:

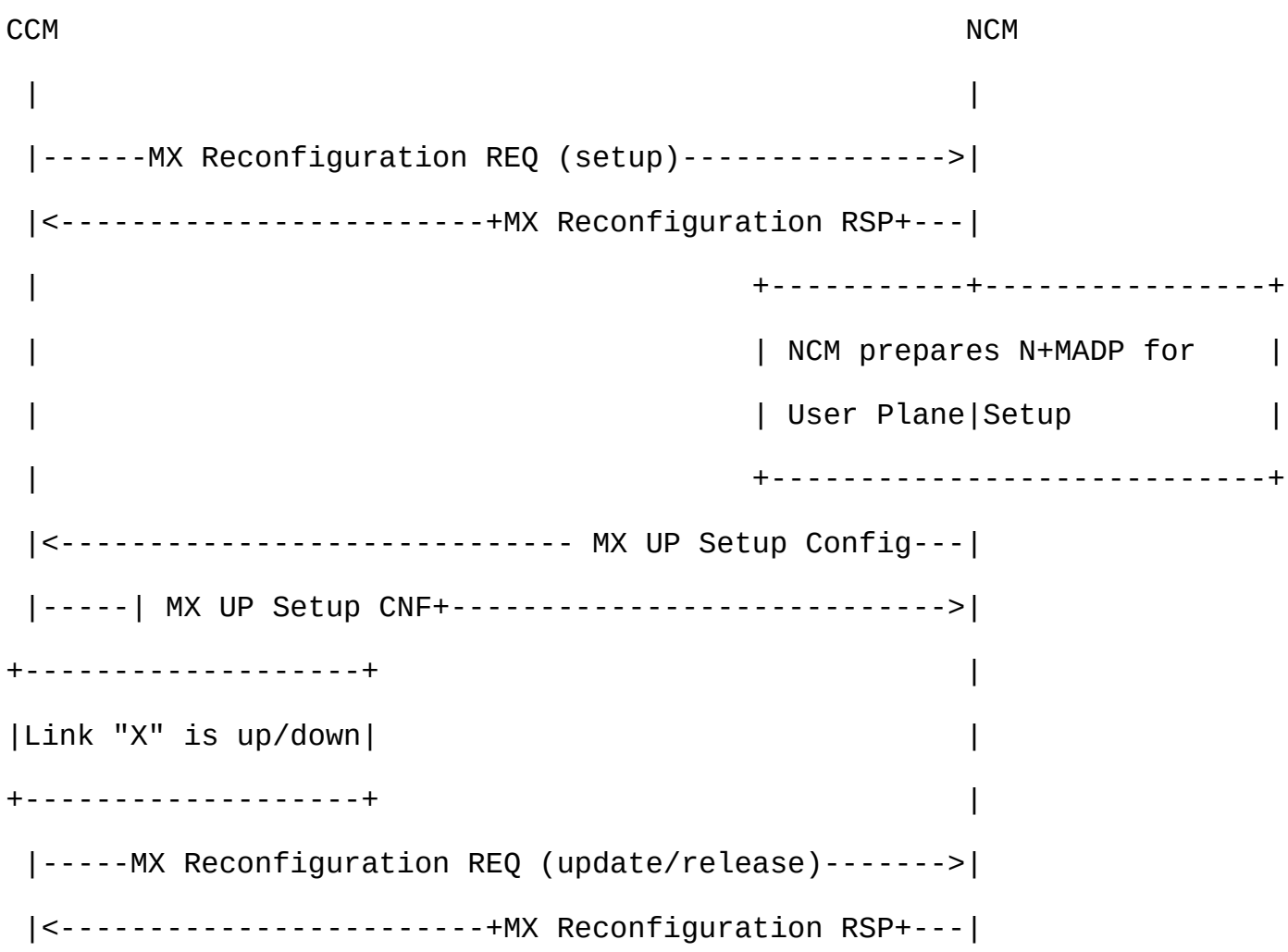

Figure 6: MAMS Control Procedure for User Plane Configuration

Reconfiguration: when the client detects that the link is up/down or the IP address changes (e.g. via APIs provided by the client OS), CCM sends out a MX Reconfiguration REQ Message to setup / release / update the connection, and the message SHOULD include the following information

- Unique Session Identifier: Identity of the CCM identity at NCM, created by NCM during the capability exchange phase.

- Reconfiguration Action: indicate the reconfiguration action (0:release; 1: setup; 2: update).
- Connection ID: identify the connection for reconfiguration

If (Reconfiguration Action is setup or update), then include the following parameters

- IP address of the connection
- SSID (if Connection Type = WiFi)
- MTU of the connection: MTU of the delivery path that is calculated at the UE for use by NCM to configure fragmentation and concatenation procedures[I-D.zhu-intarea-mams-user-protocol] at N-MADP.
- Delivery Node Identity: Identity of the node to which the client is attached. ECGI in case of LTE and WiFi AP Id or MAC address in case of WiFi.

At the beginning of a connection setup, CCM informs the NCM of the connection status using the MX Reconfiguration REQ message with Reconfiguration Action type set to "setup". NCM acknowledges the connection setup status and exchanges parameters with the CCM for user plane setup, described as follows.

User Plane Protocols Setup: Based on the negotiated capabilities, NCM sets up the user plane (Adaptation Layer and Convergence Layer) protocols at the N-MADP, and informs the CCM of the user plane protocols to setup at the client (C-MADP) and the parameters for C-MADP to connect to N-MADP.

The MX UP Setup Config is used to create (multiple) MADP instances with each Anchor Connection having one or more Configurations, namely MX Configurations. It consists of the following parameters:

- Number of Anchor Connections (Core Networks)

  For Each Anchor Connection, it includes the following parameters

  * Anchor Connection ID
  * Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
  * Number of Active MX Configurations (Included only if more than one MX configurations are active for the anchor connection)

    For each active MX configuration, it includes the following parameters

    + MX Configuration ID (included if more than one MX Configuration is present

+ MX Convergence Method, one of the following

  - GMA
  - MPTCP Proxy
  - GRE Aggregation Proxy
  - MPQUIC
+ MX Convergence Method Parameters

  - Convergence Proxy IP Address
  - Convergence Proxy Port
  - Client Key
+ MX Convergence Control Parameters (included if any MX Control PDU, e.g. Probe-REQ/ACK, is supported):

  - UDP Port Number for sending and receiving MX Control PDUs, e.g. Probe-REQ/ACK, Keep-Alive, etc.)
  - Convergence Proxy Port
+ Number of Delivery Connections

  For each Delivery Connection, include the following:

  - Delivery Connection ID
  - Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
  - MX Adaptation Method, one of the following

    o UDP Tunnel without DTLS
    o UDP Tunnel with DTLS
    o IPSec Tunnel
    o Client NAT
  - MX Adaptation Method Parameters

    o Tunnel Endpoint IP Address
    o Tunnel Endpoint Port
    o Shared Secret
    o Header Optimization (included only if MX Convergence Method is GMA)

e.g. When LTE and Wi-Fi are the two user plane accesses, NCM conveys to CCM that IPsec needs to be setup as the MX Adaptation Layer over the Wi-Fi Access, using the following parameters - IPsec end-point IP address, Pre-Shared Key. No Adaptation Layer is needed or Client NAT may be used over the LTE Access as it is considered secure with no NAT.

Similarly, as an example of the MX Convergence Method configuration is to indicate the convergence protocol as MPTCP Proxy along with

parameters for connection to the MPTCP Proxy, namely IP Address and Port of the MPTCP Proxy for TCP Applications.

Once the user plane protocols are configured, CCM informs the NCM of the status via the MX UP Setup CNF message. The MX UP Setup CNF consists of the following parameters:

- Unique Session Identifier: Session identifier provided to the client in MX Capability RSP.
- MX Convergence Control Parameters (included if any MX Control PDU, e.g. Probe-REQ/ACK, Keep-alive, is supported):
  - UDP Port Number for sending and receiving MX Control PDUs, e.g. Probe-REQ/ACK, Keep-Alive, etc.)
  - MX Configuration ID (if MX Configuration ID is specified in MX UP Setup Config, indicate the MX Configuration that will be used for Probing)
- Client Adaptation Layer Parameters:
  - Number of Delivery Connections
  - For each Delivery Connection, include the following:
    - Delivery Connection ID
    - UDP port number: If UDP based adaptation is in use, the UDP port at C-MADP side

8.6. MAMS Path Quality Estimation

Path quality estimations can be done either passively or actively. Traffic measurements in the network could be performed passively by comparing the real-time data throughput of the device with the capacity available in the network. In special deployments where the NCM has interfaces with access nodes, direct interfaces can be used to gather path quality information. For example, the utilization of a cell/eNB attached to a device could be used as an indicator for path quality estimations without creating an extra traffic overhead. Active measurements by the device are an alternative for estimating path quality.

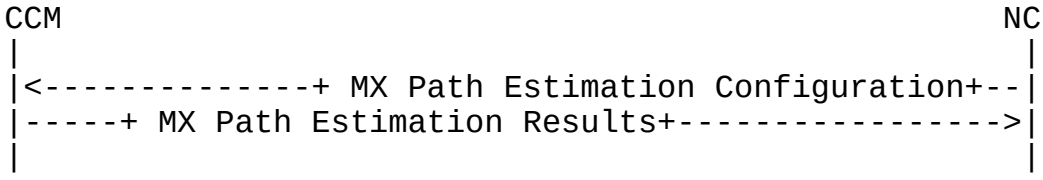

Figure 7: MAMS Control Plane Procedure for Path Quality Estimation

NCM sends following the configuration parameters in the MX Path Estimation Configuration message to the CCM

- Connection ID (of Delivery Connection whose path quality needs to be estimated)
- Init Probe Test Duration (ms)
- Init Probe Test Rate (Mbps)
- Init Probe Size (Bytes)
- Init Probe Ack Required (0 -> No/1 -> Yes)
- Active Probe Frequency (ms)
- Active Probe Size (Bytes)
- Active Probe Test Duration (ms)
- Active Probe Ack Required (0 -> No/1 -> Yes)

CCM configures the C-MADP for probe reception based on these parameters and for collection of the statistics according to the following configuration.

- Unique Session Identifier: Session identifier provided to the client in MX Capability RSP.
- Init Probe Results Configuration
  - Lost Probes (%)
  - Probe Receiving Rate (packets per second)
- Active Probe Results Configuration
  - Average Throughput in the last Probe Duration

The user plane probing is divided into two phases - Initialization phase and Active phase.

- Initialization phase: A network path that is not included by N-MADP for transmission of user data is deemed to be in the Initialization phase. The user data may be transmitted over other available network paths.

- Active phase: A network path that is included by N-MADP for transmission of user data is deemed to be in Active phase.

In Initialization phase, NCM configures N-MADP to send an MX Idle Probe REQ message. CCM collects the Idle probe statistics from C-MADP and sends the MX Path Estimation Results Message to NCM per the Initialization Probe Results configuration.

In Active phase, NCM configures N-MADP to send an MX Active Probe REQ message.. C-MADP calculates the metrics as specified by the Active Probe Results Configuration. CCM collects the Active probe statistics from C-MADP and sends the MX Path Estimation Results Message to NCM per the Active Probe Results configuration.

The following sub-sections define the control PDU encoding for Probe and Keep Alive messages to support path quality estimation.

8.6.1. MX Control PDU definition

Control PDUs are sent as UDP Messages between C-MADP and N-MADP to exchange control messages for keep-alive or path quality estimation. MX Probe Parameters are negotiated during the User Plane Setup phase (MX UP SETUP CFG and MX UP SETUP CNF). Figure 7 shows the MX control PDU format with the following fields:

- Type (1 Byte): the type of the MX control message
  - 0: Keep-Alive
  - 1: Probe REQ/ACK
  - Others: Reserved
- CID (1 Byte): the connection ID of the delivery connection for sending out the MX control message
- MX Control Message (variable): the payload of the MX control message
- Figure 8 shows the MX Control PDU format. MX Control PDU is sent as a normal user plane packet over the desired delivery connection whose quality and reachability needs to be determined.

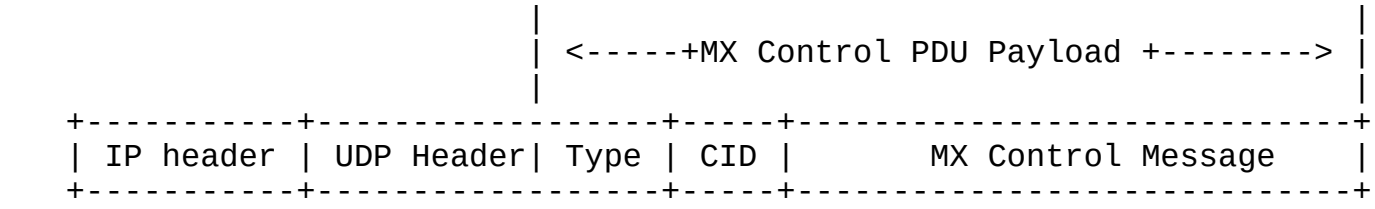

Figure 8: MX Control PDU Format

8.6.2. Keep-Alive Message

The "Type" field is set to "0" for Keep-Alive messages. C-MADP may send out Keep-Alive message periodically over one or multiple delivery connections, especially if UDP tunneling is used as the adaptation method for the delivery connection with a NAT function on the path.

A Keep-Alive message is 2 Bytes long, and consists of the following fields:

- Keep-Alive Sequence Number (2 Bytes): the sequence number of the keep-alive message.

8.6.3. Probe REQ/ACK Message

The "Type" field is set to "1" for Probe REQ/ACK messages. N-MADP may send out the Probe REQ message for path quality estimation. In response, C-MADP may send back the Probe ACK message.

A Probe REQ message consists of the following fields:

- Probing Sequence Number (2 Bytes): the sequence number of the Probe REQ message
- Probing Flag (1 Byte):
  - Bit #0: a Probe ACK flag to indicate if the Probe ACK message is expected (1) or not (0);
  - Bit #1: a Probe Type flag to indicate if the Probe REQ/ACK message is sent during the initialization phase (0) when the network path is not included for transmission of user data or the active phase (1) when the network path is included for transmission of user data;
  - Bit #2: a bit flag to indicate the presence of the Reverse Connection ID (R-CID) field.
  - Bit #3~7: reserved

- Reverse Connection ID (1 Byte): the connection ID of the delivery connection for sending out the Probe ACK message on the reverse path
- Padding (variable)

The "R-CID" field is only present if both Bit #0 and Bit #2 of the "Probing Flag" field are set to "1". Moreover, Bit #2 of the "Probing Flag" field SHOULD be set to "0" if the Bit #0 is "0", indicating the Probe ACK message is not expected.

If the "R-CID" field is not present but the Bit #0 of the "Probing Flag" field is set to "1", the Probe ACK message SHOULD be sent over the same delivery connection as the Probe REQ message.

The "Padding" field is used to control the length of Probe REQ message.

C-MADP SHOULD send out the Probe ACK message in response to a Probe REQ message with the Probe ACK flag set to "1".

A Probe ACK message is 3 Bytes long, and consists of the following fields:

- Probing Acknowledgement Number (2 Bytes): the sequence number of the corresponding Probe REQ message

8.7. MAMS Traffic Steering

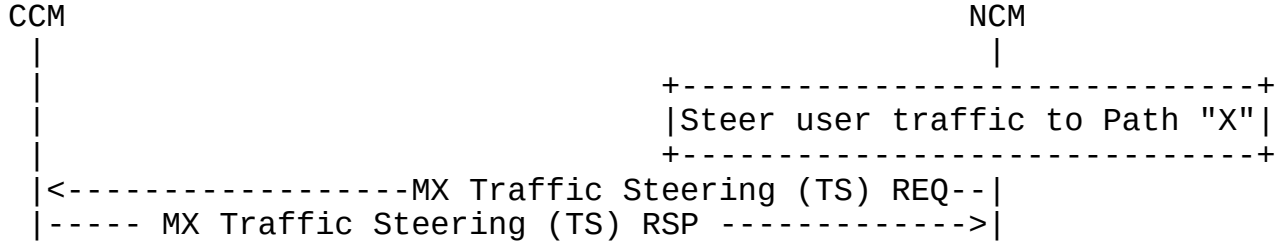

Figure 9: MAMS Traffic Steering Procedure

NCM sends out a MX Traffic Steering (TS) REQ message to steer data traffic. It is also possible to send data traffic over multiple connections simultaneously, i.e. aggregation. The message includes the following information:

- Connection ID of the Anchor Connection
- MX Configuration ID (if MX Configuration ID is specified in MX UP Setup Config)

- o Connection ID List of Delivery Connections for DL traffic
- o Connection ID of Default UL Delivery Connection
- o For the number of Specific UL traffic Templates, include the following
  - * Traffic Template for identifying the UL traffic
  - * Connection ID List of Delivery connections for UL traffic identified by the traffic template
- o MX Feature Activation List: each parameter indicates if the corresponding feature is enabled or not: lossless switching, fragmentation, concatenation, Uplink aggregation, Downlink aggregation, Measurement, probing

In response, CCM sends out a MX Traffic Steering (TS) RSP message, including the following information:

- o Unique Session Identifier: Session identifier provided to the client in MX Capability RSP.
- o MX Feature Activation List: each parameter indicates if the corresponding feature is enabled or not: lossless switching, fragmentation, concatenation, Uplink aggregation, Downlink aggregation, probing

8.8. MAMS Application MADP Association

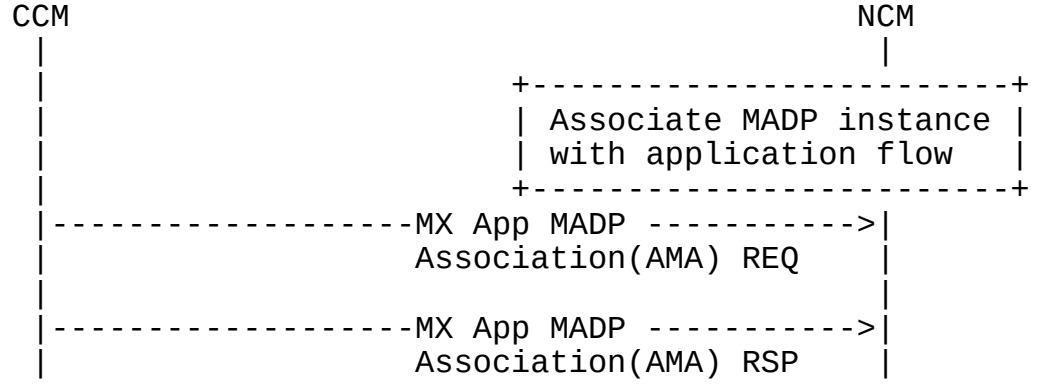

Figure 10: MAMS Application MADP Association Procedure

CCM sends out a MX App MADP Association(AMA) REQ message to request association of a specific Application flow with a specific MADP instance ID for the anchor connection with multiple active MX configurations. MADP Instance ID is a tuple (Anchor Connection ID, MX Configuration ID). This provides the capability for the client to indicate the user plane processing that needs to be associated with different application flows depending on their needs. The

application flow is identified by its associated traffic flow template.

The message includes the following information:

- Number of Application Flows

  For Each Application Flow, identified by the Traffic Flow Template(s),

  * Anchor Connection ID
  * MX Configuration ID (if more than one MX Configurations are associated with an Anchor Connection)
  * Traffic Template for identifying the UL traffic
  * Traffic Template for identifying the DL traffic

In response, NCM sends out a MX App MADP Association (AMA) RSP message, including the following information:

- Number of Application Flows

  For Each Application Flow, identified by the Traffic Flow Template(s),

  * Status (Success or Failure)

8.9. MAMS Network ID Indication

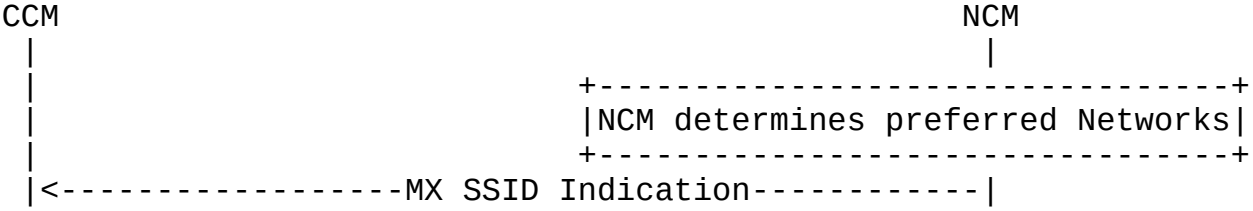

Figure 11: MAMS Network ID Indication Procedure

NCM indicates the preferred network list to the CCM to guide client on networks that it should connect to. To indicate preferred Wi-Fi Networks, the NCM sends the list of WLAN networks, represented by SSID/BSSID/HESSID, available in the MX SSID Indication.

8.10. MAMS Client Measurement Configuration and Reporting

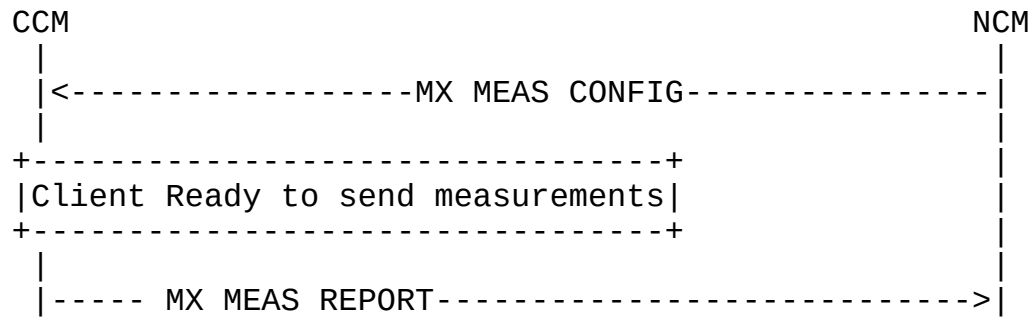

Figure 12: MAMS Client Measurement Configuration and Reporting Procedure

NCM configures the CCM with the different parameters (e.g. radio link information), with the associated thresholds to be reported by the client. The MX MEAS CONFIG message contains the following parameters. For each Delivery Connection, include the following:

- Delivery Connection ID
- Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
- If Connection Type is Wi-Fi
  - WLAN_RSSI_THRESH: High and Low Thresholds for sending Average RSSI of the Wi-Fi Link.
  - WLAN_RSSI_PERIOD: Periodicity in ms for sending Average RSSI of the Wi-Fi Link.
  - WLAN_LOAD_THRESH: High and Low Thresholds for sending Loading of the WLAN system.
  - WLAN_LOAD_PERIOD: Periodicity in ms for sending Loading of the WLAN system.
  - UL_TPUT_THRESH: High and Low Thresholds for sending Reverse Link Throughput on the Wi-Fi link.
  - UL_TPUT_PERIOD: Periodicity in ms for sending Reverse Link Throughput on the Wi-Fi link.
  - DL_TPUT_THRESH: High and Low Thresholds for sending Forward Link Throughput on the Wi-Fi link.
  - DL_TPUT_PERIOD: Periodicity in ms for sending Forward Link Throughput on the Wi-Fi link.
  - EST_UL_TPUT_THRESH: High and Low Thresholds for sending Reverse Link Throughput (EstimatedThroughputOutbound as defined in [IEEE]) on the Wi-Fi link.

    * EST_UL_TPUT_PERIOD: Periodicity in ms for sending Reverse Link Throughput (EstimatedThroughputOutbound as defined in [IEEE]) on the Wi-Fi link.
    * EST_DL_TPUT_THRESH: High and Low Thresholds for sending Forward Link Throughput (EstimatedThroughputInbound as defined in [IEEE]) on the Wi-Fi link.
    * EST_DL_TPUT_PERIOD: Periodicity in ms for sending Forward Link Throughput (EstimatedThroughputInbound as defined in [IEEE]) on the Wi-Fi link.

  o If Connection Type is LTE

    * LTE_RSRP_THRESH: High and Low Thresholds for sending RSRP of Serving LTE link.
    * LTE_RSRP_PERIOD: Periodicity in ms for sending RSRP of Serving LTE link.
    * LTE_RSRQ_THRESH: High and Low Thresholds for sending RSRQ of the serving LTE link.
    * LTE_RSRQ_PERIOD: Periodicity in ms for sending RSRP of Serving LTE link.
    * UL_TPUT_THRESH: High and Low Thresholds for sending Reverse Link Throughput on the serving LTE link.
    * UL_TPUT_PERIOD: Periodicity in ms for sending Reverse Link Throughput on the serving LTE link.
    * DL_TPUT_THRESH: High and Low Thresholds for sending Forward Link Throughput on the serving LTE link.
    * DL_TPUT_PERIOD: Periodicity in ms for sending Forward Link Throughput on the serving LTE link.

  o If Connection Type is 5G NR

    * NR_RSRP_THRESH: High and Low Thresholds for sending RSRP of Serving NR link.
    * NR_RSRP_PERIOD: Periodicity in ms for sending RSRP of Serving NR link.
    * NR_RSRQ_THRESH: High and Low Thresholds for sending RSRQ of the serving NR link.
    * NR_RSRQ_PERIOD: Periodicity in ms for sending RSRP of Serving NR link.
    * UL_TPUT_THRESH: High and Low Thresholds for sending Reverse Link Throughput on the serving NR link.
    * UL_TPUT_PERIOD: Periodicity in ms for sending Reverse Link Throughput on the serving NR link.
    * DL_TPUT_THRESH: High and Low Thresholds for sending Forward Link Throughput on the serving NR link.
    * DL_TPUT_PERIOD: Periodicity in ms for sending Forward Link Throughput on the serving NR link.

  The MX MEAS REPORT message contains the following parameters

- Unique Session Identifier: Session identifier provided to the client in MX Capability RSP.
- For each Delivery Connection, include the following:
  - Delivery Connection ID
  - Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
  - Delivery Node Identity (ECGI in case of LTE and WiFi AP Id or MAC address in case of WiFi)
  - If Connection Type is Wi-Fi
    - WLAN_RSSI: Average RSSI of the Wi-Fi Link.
    - WLAN_LOAD: Loading of the WLAN system.
    - UL_TPUT: Reverse Link Throughput on the Wi-Fi link.
    - DL_TPUT: Forward Link Throughput on the Wi-Fi link.
    - EST_UL_TPUT: Estimated Reverse Link Throughput on the Wi-Fi link (EstimatedThroughputOutbound as defined in [IEEE]).
    - EST_DL_TPUT: Estimated Forward Link Throughput on the Wi-Fi link (EstimatedThroughputInbound as defined in [IEEE]).
  - If Connection Type is LTE
    - LTE_RSRP: RSRP of Serving LTE link.
    - LTE_RSRQ: RSRQ of the serving LTE link.
    - UL_TPUT: Reverse Link Throughput on the serving LTE link.
    - DL_TPUT: Forward Link Throughput on the serving LTE link.
  - If Connection Type is 5G NR
    - NR_RSRP: RSRP of Serving NR link.
    - NR_RSRQ: RSRQ of the serving NR link.
    - UL_TPUT: Reverse Link Throughput on the serving NR link.
    - DL_TPUT: Forward Link Throughput on the serving NR link.

8.11. MAMS Session Termination Procedure

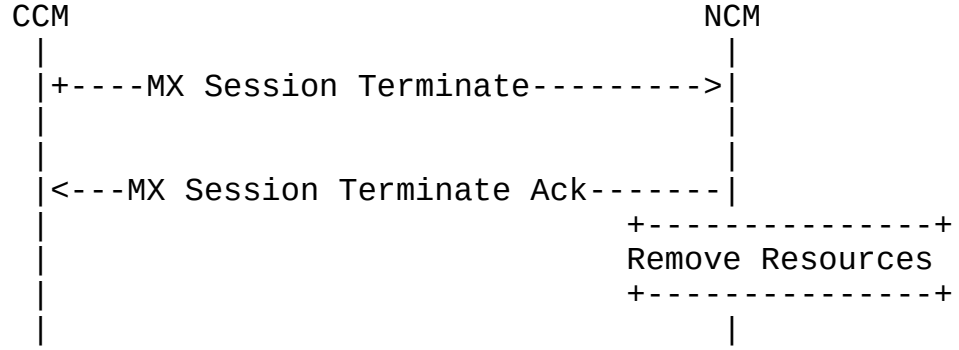

Figure 13: MAMS Session Termination Procedure - Client Initiated

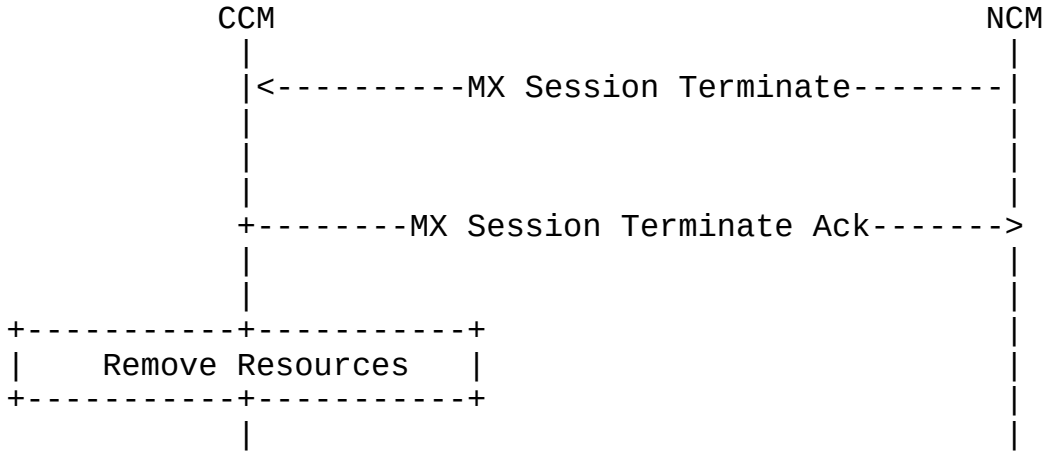

Figure 14: MAMS Session Termination Procedure - Network Initiated

At any point in MAMS functioning if CCM or NCM is unable to support the MAMS functions anymore, then either of them can initiate a termination procedure by sending MX Session Terminate to the peer, the peer shall acknowledge the termination by sending MX Session Terminate ACK message. After the session is disconnected the CCM shall start a new procedure with MX Discover Message. MX Session Terminate message shall contain Unique Session Identifier and reason for termination in Request. Possible reasons for termination can be:

- Normal Release
- No Response from Peer
- Internal Error

8.12. MAMS Network Analytics Request Procedure

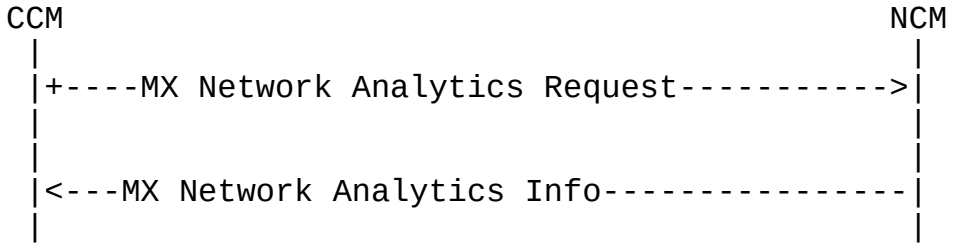

Figure 15: MAMS Network Analytics Request Procedure

CCM sends the MX Network Analytics Request informs the NCM to send information related to network parameters like bandwidth, latency, jitter, signal quality based on application of analytics at the

network utilizing the received path measurements and client measurement reporting.

The MX Network Analytics Request message consists of the following parameters.

- Link Quality Indicators, one or more of the following:
  - Bandwidth
  - Jitter
  - Latency
  - Signal Quality

NCM sends the MX Network Analytics Info to convey the analytics info, predictive parameters with likelihoods, for the different parameters of interest for the CCM.

The MX Network Analytics Info messages consists of the following parameters.

- Number of Delivery Connections For Each Delivery Connection,
  - Access Link Identifier
    - Connection Type
    - Connection ID
  - Link Quality Indicator
    - Bandwidth
      - Predicted Value (in Mbps)
      - - Likelihoood (in Percentage)
      - Prediction Validity (Validity Time in s)
    - Jitter
      - Predicted Value (in s)
      - - Likelihoood (in Percentage)
      - Prediction Validity (Validity Time in s)
    - Latency
      - Predicted Value (in s)
      - - Likelihoood (in Percentage)
      - Prediction Validity (Validity Time in s)
    - Signal Quality
      - if Delivery Connection Type is LTE, LTE_RSRP Predicted Value (in dBm)

- if Delivery Connection Type is LTE, LTE_RSRQ Predicted Value (in dBm)
- if Delivery Connection Type is NR, NR_RSRP Predicted Value (in dBm)
- if Delivery Connection Type is NR, NR_RSRQ Predicted Value (in dBm)
- if Delivery Connection Type is WiFi, WLAN_RSSI Predicted Value (in dBm)
- - Likelihoood (in Percentage)
- Prediction Validity (Validity Time in s)

9. Generic MAMS Signaling Flow

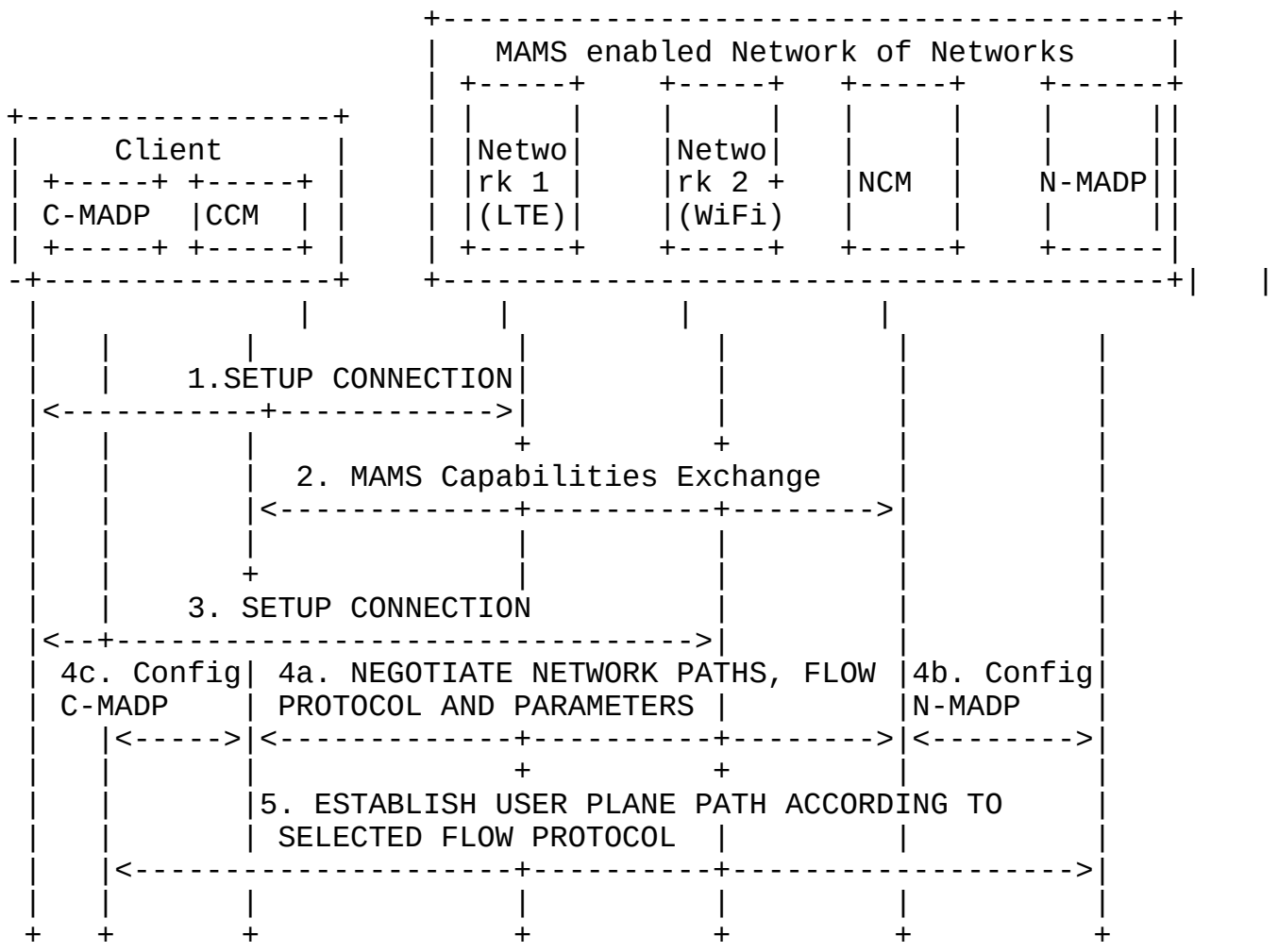

Figure 16: MAMS call flow

Figure 16 illustrates the MAMS signaling mechanism for negotiation of network paths and flow protocols between the client and the network. In this example scenario, the client is connected to two networks (say LTE and WiFi).

1. UE connects to network 1 and gets an IP address assigned by network 1.
2. CCM communicates with NCM functional element via the network 1 connection and exchanges capabilities and parameters for MAMS operation. Note: The NCM credentials (e.g. NCM IP Address) can be made known to the UE by pre-provisioning.
3. Client sets up connection with network 2 and gets an IP address assigned by network 2.
4. CCM and NCM negotiate capabilities and parameters for establishment of network paths, which are then used to configure user plane functions N-MADP at the network and C-MADP at the client.

   4a. CCM and NCM negotiate network paths, flow routing and aggregation protocols, and related parameters.

   4b. NCM communicates with the N-MADP to exchange and configure flow aggregation protocols, policies and parameters in alignment with those negotiated with the CCM.

   4c. CCM communicates with the C-MADP to exchange and configure flow aggregation protocols, policies and parameters in alignment with those negotiated with the NCM.

5. C-MADP and N-MADP establish the user plane paths, e.g. using IKE [RFC7296] signaling, based on the negotiated flow aggregation protocols and parameters specified by NCM.

CCM and NCM can further exchange messages containing access link measurements for link maintenance by the NCM. NCM evaluates the link conditions in the UL and DL across LTE and WiFi, based on link measurements reported by CCM and/or link probing techniques and determines the UL and DL user data distribution policy. NCM and CCM also negotiate application level policies for categorizing applications, e.g. based on DSCP, Destination IP address, and determining which of the available network paths, needs to be used for transporting data of that category of applications. NCM configures N-MADP, and CCM configures C-MADP, based on the negotiated application policies. CCM may apply local application policies, in addition to the application policy conveyed by the NCM.

10. Relation to IETF Technologies

MAMS leverages technologies developed in IETF like MPTCP, GRE and enables a control plane framework to negotiate the use of these protocols between the client and the network. It also addresses the limitations in the scope of other multihoming protocols. E.g. MOBIKE RFC 4555 (IKEv2 Mobility and Multihoming Protocol (MOBIKE)) scope indicates that it is limited to multihoming between IPsec clients(tunnel mode IPsec SAs) ONLY and does NOT support load balancing. MAMS addresses this limitation in handling multihoming scenario by supporting load balancing with simultaneous use of multiple access paths by negotiating use of protocols like MPTCP. Unlike MOBIKE, which only applies to end points connected with IPsec tunnel mode SA, MAMS allows flexibility to use a wide range of tunneling protocols to be used in the Adaptation layer.

11. Applying MAMS Control Procedures with MPTCP Proxy as User Plane

If NCM determines that N-MADP is to be instantiated with MPTCP as the MX Convergence Protocol, it exchanges the MPTCP capability support in discovery and capability exchange procedures. NCM then exchanges the credentials of the N-MADP instance, setup as MPTCP Proxy, along with related parameters to the CCM. CCM configures C-MADP with these parameters to connect with the N-MADP, MPTCP proxy (e.g. [I-D.ietf-tcpm-converters]) instance, on the available network path (Access).

Figure 17 illustrates the user plane protocol layering when MPTCP is configured to be the "MX Convergence Sublayer" protocol. MPTCP manages traffic distribution and aggregation over multiple delivery connections.

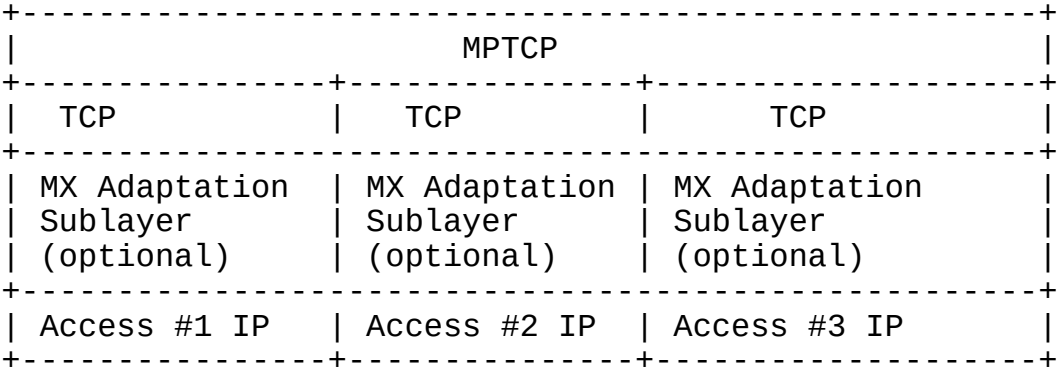

Figure 17: MAMS U-plane Protocol Stack with MPTCP as MX Convergence Layer

The Client (C-MADP) sets up an MPTCP connection with the N-MADP to begin with. MAMS control procedures are then applied to,

- Connect to the appropriate MPTCP network endpoint, e.g. MPTCP Proxy (illustrated in Figure 18)
- Control addition of second TCP subflow after WiFi connection is established and is deemed good, (illustrated in illustrated in Figure 19),
- Control the MPTCP scheduler behavior like use of only LTE Subflow in UL and both LTE and WiFi subflows in DL (illustrated in illustrated in Figure 20),
- Faster response to WiFi link degradation by proactive deletion of TCP subflow over WiFi when poor link conditions are reported, to maintain performance (illustrated in illustrated in Figure 21)

Figure 18 shows the call flow describing MAMS control procedures applied to configure user plane and dynamic optimal path selection in a scenario with MPTCP Proxy as the convergence protocol in the user plane.

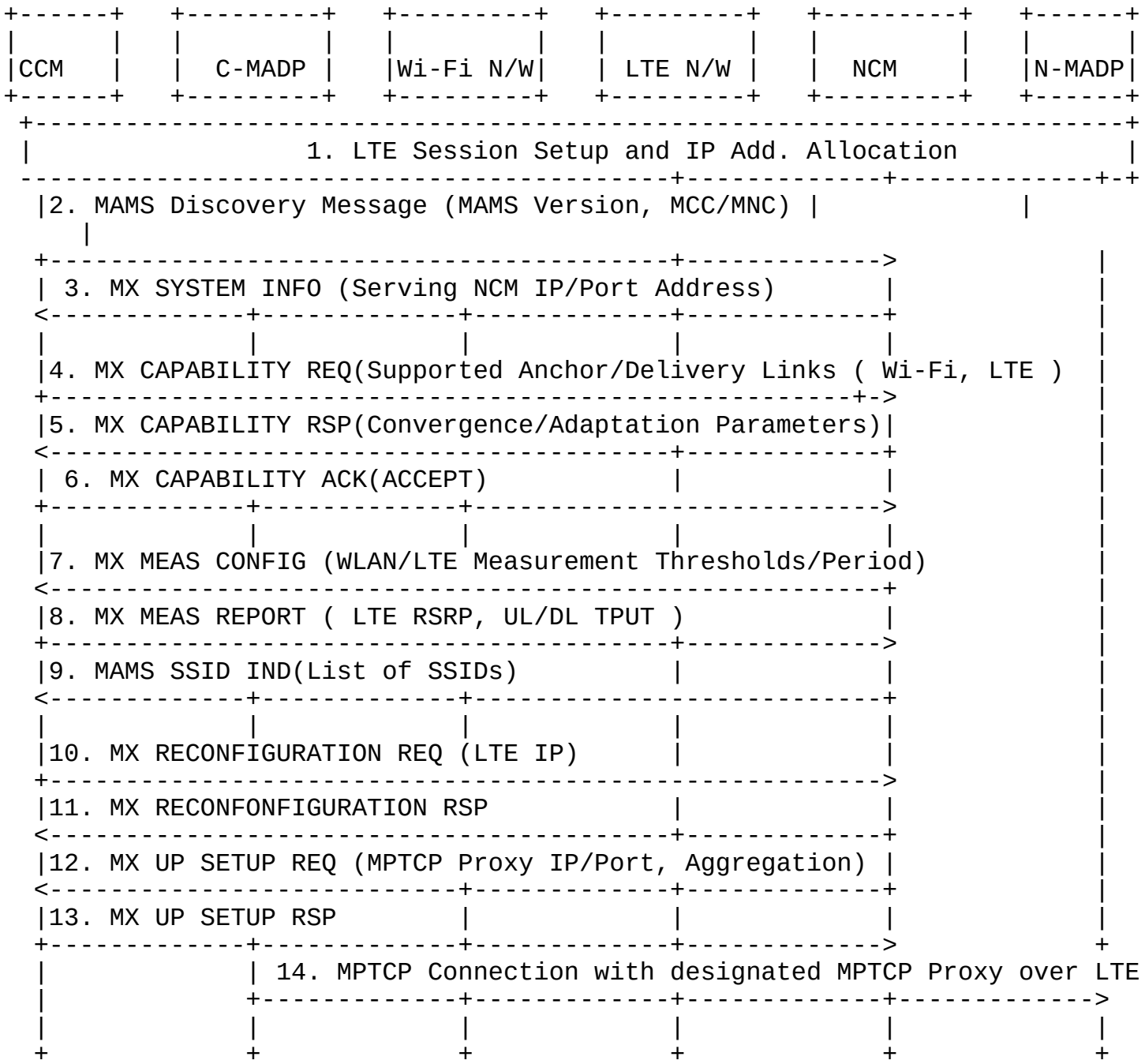

Figure 18: MAMS-assisted MPTCP Proxy as User Plane - Initial Setup with LTE leg

Following are the salient steps described in the call flow. The client connects to the LTE network and obtains an IP address (assume LTE is the first connection), and initiates the NCM discovery procedures and exchange capabilities, including the support for MPTCP as the convergence protocol at both the network and the client.

The CCM informs the LTE connection parameters to the NCM. NCM provides the parameters like MPTCP Proxy IP address/Port, MPTCP Client Key for configuring the convergence layer. This is useful if N-MADP is reachable via different IP address or/and port, from different access networks. The current MPTCP signaling can’t identify or differentiate the MPTCP proxy IP address and port among multiple access networks. The client uses the MPTCP Client Key during the subflow creation, and this enables the N-MADP to uniquely identify the client, even if NAT is present. The N-MADP then can inform the NCM of the subflow creation and pararmeters related to creating additional subflows. Since LTE is the only connection, the user plane traffic, flows over the single TCP subflow over the LTE connection. Optionally, NCM provides assistance information to the device on the neighboring/preferred Wi-Fi networks that it can associate with.

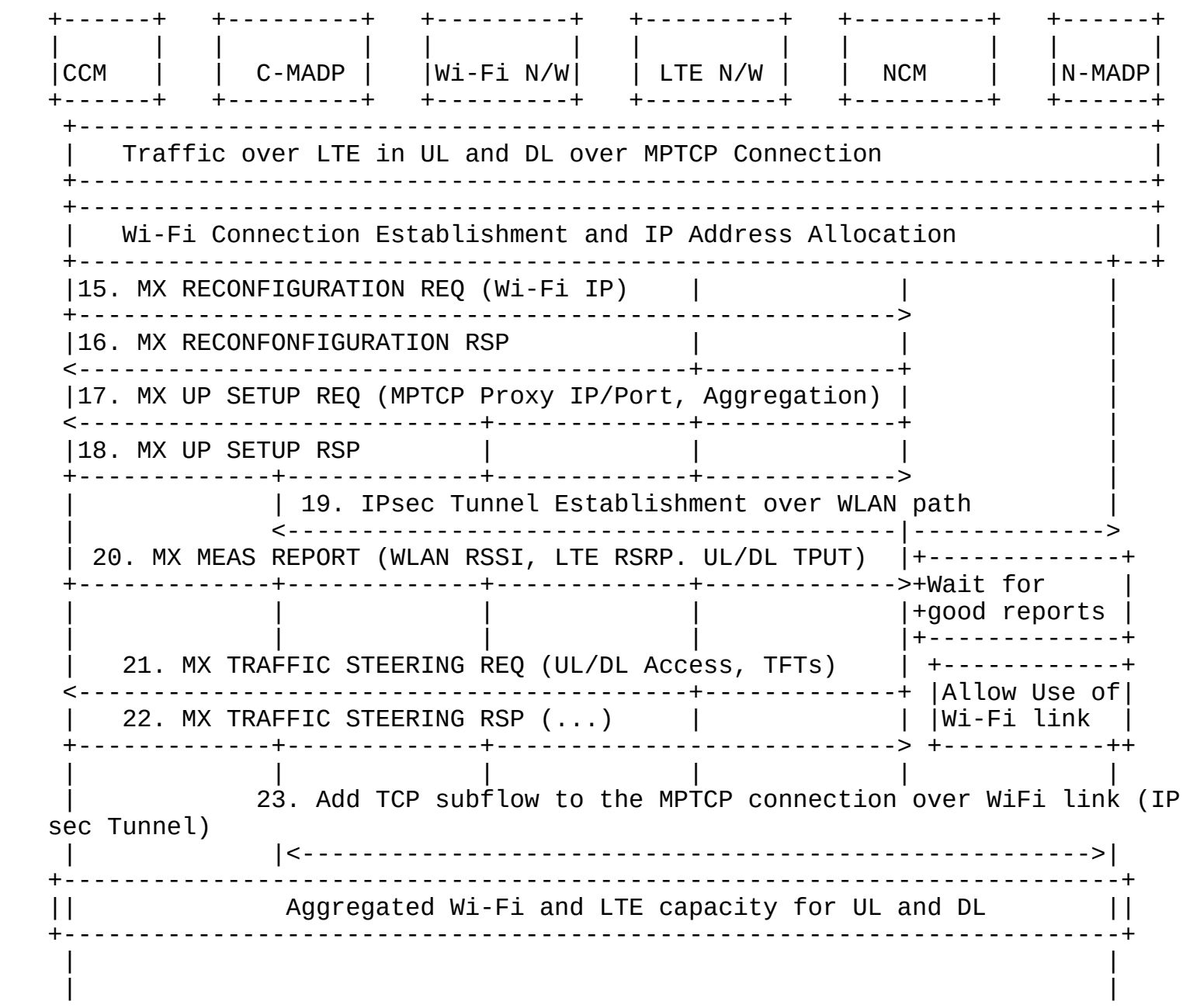

Figure 19: MAMS-assisted MPTCP Proxy as User Plane - Add Wi-Fi leg

Figure 19 describes the steps, when the client establishes a Wi-Fi connection. CCM informs the NCM of the Wi-Fi connection along with parameters like the Wi-Fi IP address, SSID. NCM determines that the Wi-Fi connection needs to be secured and configures the Adaptation Layer to be IPsec and provides the required parameters to the CCM. In addition, NCM provides the information to configure the convergence layer, (e.g. MPTCP Proxy IP Address), and provides the Traffic Steering Request to indicate that client should use only the

LTE access. NCM may do this, for example, on determination from the measurements that the Wi-Fi link is not consistently good enough. As the Wi-Fi link conditions improve, NCM sends a Traffic Steering Request to use Wi-Fi access as well. This triggers the client to establish the TCP subflow over the Wi-Fi link with the MPTCP proxy

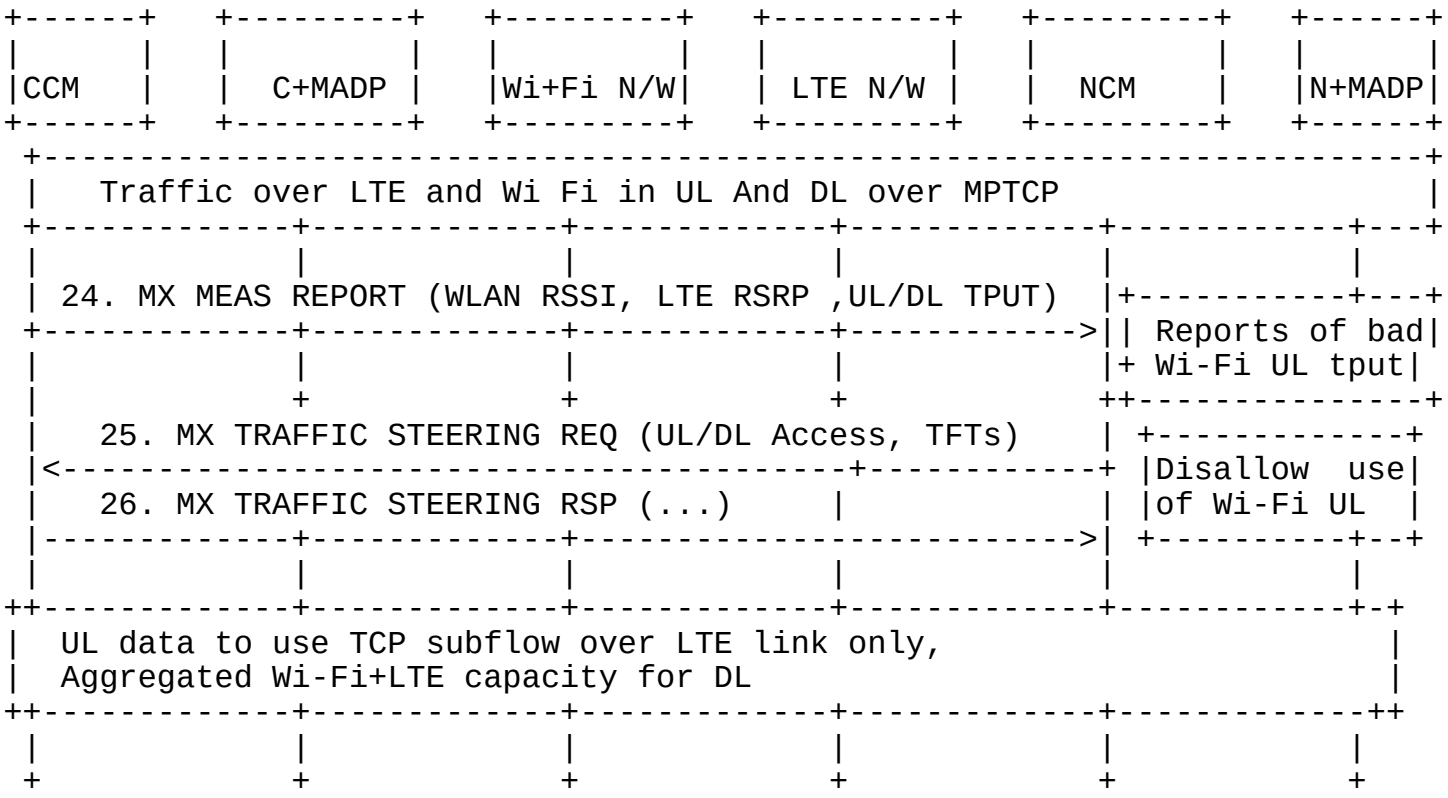

Figure 20: MAMS-assisted MPTCP Proxy as User Plane - Wi-Fi UL degrades

Figure 20 describes the steps, when the client reports that Wi-Fi link conditions degrade in UL. MAMS control plane is used to continuously monitor the access link conditions on Wi-Fi and LTE connections. The NCM may at some point determine increase in UL traffic on Wi-Fi, and trigger the client to only LTE in the UL via Traffic Steering Request to improve UL performance.

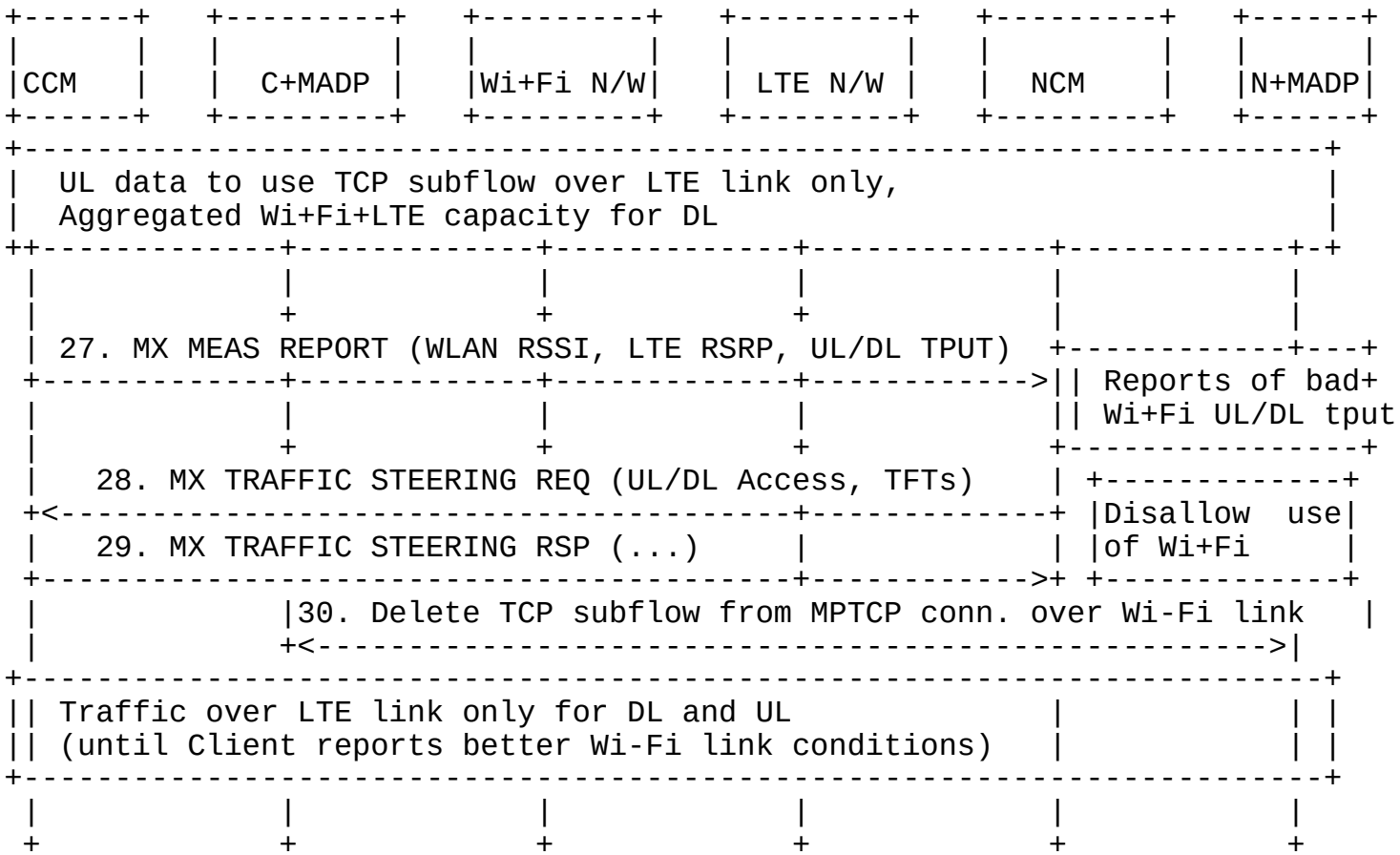

Figure 21: MAMS-assisted MPTCP Proxy as User Plane - Part 4

Figure 21 describes the steps, when the client reports that Wi-Fi link conditions degrade in both UL and DL. As the Wi-Fi link conditions deteriorate further, the NCM may determine to send Traffic Steering Request guiding the client to stop using Wi-Fi, and to use only LTE access in both UL and DL. This condition may be maintained until NCM determines, based on reported measurements that Wi-Fi link has become usable.

12. Applying MAMS Control Procedures for Network Assisted Traffic Steering when there is No Convergence Layer

Figure 22 shows the call flow describing MAMS control procedures applied for dynamic optimal path selection in a scenario convergence and Adaptation layer protocols are not omitted. This scenario indicates the applicability of a MAMS Control Plane only solution.

In the capability exchange messages, NCM and CCM negotiate that Convergence and Adaptation layer protocols are not needed (or

supported). CCM informs the NCM of the availability of the LTE and Wi-Fi links. NCM determines the access links, Wi-Fi or LTE to be used dynamically based on the reported link quality measurements.

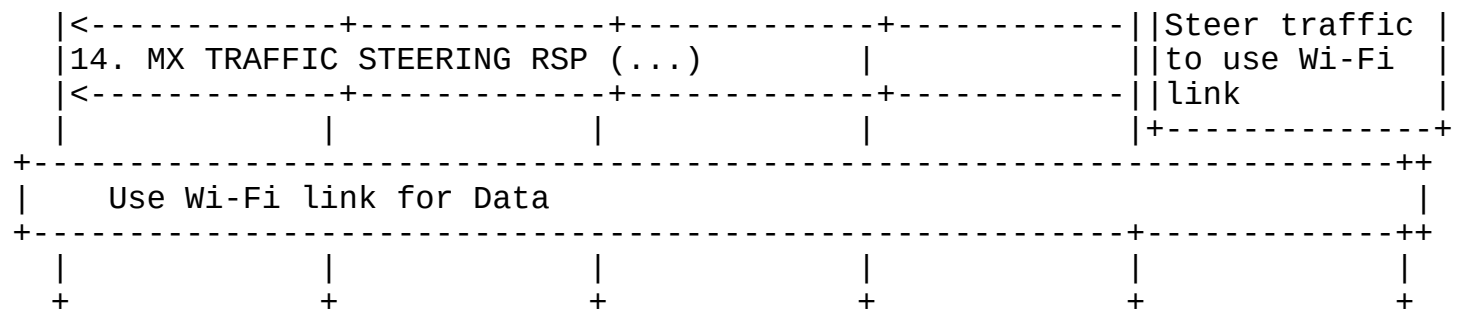

Figure 22: MAMS With No Convergence Layer

13. Co-existence of MX Adaptation and MX Convergence Layers

MAMS user plane supports multiple instances and combinations of protocols to be used at the MX Adaptation and the Convergence layer.

For example, one instance of the MX Convergence Layer can be MPTCP Proxy and another instance can be GMA. The MX Adaptation for each can be either UDP tunnel or IPsec. IPSec may be set up when network path needs to be secured, e.g. to protect the TCP subflow traversing the network path between the client and MPTCP proxy.

Each of the instances of MAMS user plane, i.e. combination of MX Convergence and MX Adaptation layer protocols, can coexist simultaneously and independently handle different traffic types.

14. Security Considerations

14.1. MAMS Control Plane Security

The NCM functional element is hosted on a network node which is assumed to be within a secure network, e.g. within the operator’s network, and is assumed to be protected against hijack attacks.

For deployment scenarios, where the client is configured (e.g. by the network operator) to use a specific network path for exchanging control plane messages and if the network path is assumed to be secure, MAMS control messages will rely on security provided by the underlying network.

For deployment scenarios where the security of the network path cannot be assumed, NCM and CCM implementations MUST support the "wss" URI scheme [RFC6455] and Transport Layer Security (TLS) [RFC5246] to secure control plane message exchange between the NCM and CCM.

For deployment scenarios where client authentication is desired, the WebSocket server can use any client authentication mechanisms available to a generic HTTP server, such as cookies, HTTP authentication, or TLS authentication.

14.2. MAMS User Plane Security

User data in MAMS framework relies on the security of the underlying network transport paths. When this cannot be assumed, NCM configures use of protocols, like IPsec [RFC4301] [RFC3948] in the MX Adaptation Layer, for security.

15. Implementation Considerations

MAMS builds on commonly available functions available on terminal devices that can be delivered as a software update over the popular end-user device operating systems, enabling rapid deployment and addressing the large deployed device base.

16. Applicability to Multi Access Edge Computing

Multi-access Edge Computing (MEC), previously known as Mobile Edge Computing, is an access-edge cloud platform being considered at ETSI [ETSIMEC] , whose initial focus was to improve quality of experience by leveraging intelligence at the cellular (e.g., 3GPP technologies like LTE) access edge, and the scope is now being extended to support access technologies beyond 3GPP. The applicability of the framework described in this document to the MEC platform has been evaluated and tested in different network configurations by the authors.

The NCM can be hosted on a MEC cloud server that is located in the user plane path at the edge of the multi-technology access network. The NCM and CCM can negotiate the network path combinations based on application needs and the necessary user plane protocols to be used across the multiple paths. The network conditions reported by the CCM to the NCM can be complemented by a Radio Analytics application [ETSIRNIS] residing at the MEC to configure the uplink and downlink access paths according to changing radio and congestion conditions.

The user plane functional element, N-MADP, can either be collocated with the NCM at the MEC cloud server (e.g., MEC hosted applications), or placed at a separate network element like a common user plane gateway across the multiple networks.

Also, even in scenarios where N-MADP is not deployed, the NCM can be used to augment the traffic steering decisions at the device.

The aim of these enhancements is to improve the end-user's quality of experience by leveraging the best network path based on application needs and network conditions, and building on the advantages of significantly reduced latency and the dynamic and real-time exposure of radio network information available at the MEC.

17. Related work in other Industry and Standards Forums

The MAMS framework described in this document has been incorporated as a solution to address multi access integration in multiple industry forums and standards. This section describes the related work in other industry forums and the standards organizations.

Wireless Broadband Alliance Industry partners have published a whitepaper that describes applicability of different technologies for multi access integration to different deployments as part of their project named, Unlicensed Integration with 5G Networks [WBAUnl5G]. The whitepaper includes MAMS framework described in this document as a technology for integrating Unlicensed (WiFi) networks with 5G networks above the 5G core network.

3GPP is developing a technical report as part of its work item Study on Access Traffic Steering, Switching and Splitting (ATSSS). That report, TR23.793 [GPPATSSS], contains a number of potential solutions and Solution 1 utilizes a separate control plane for flexible negotiation of user plane protocols and path measurements in a way that is similar the MAMS architecture described in this document.

The Small Cell Forum (SCF) [SCFTECH5G] plans to develop a while paper as part of its work item on LTE/5G and WiFi. There is a proposal to include MAMS in this whitepaper.

The ETSI Multi-access Edge Computing Phase 2 technical work is examining many aspects of this work including use cases for optimizing Quality of Experience (QoE) and resource utilization. The MAMS architecture and procedures outlined in this document is included in the use cases and requirements document[ETSIMAMS].

18. Contributing Authors

The editors gratefully acknowledge the following additional contributors in alphabetical order: A Krishna Pramod/Nokia, Hannu Flinck/Nokia, Hema Pentakota/Nokia, Nurit Sprecher/Nokia, Salil Agarwal/Nokia; Shuping Peng/Huawei, Vasudevan Subramanian/Nokia. Vasudevan Subramanian has been instrumental in conceptualization and development of solution principles for the MAMS framework. Shuping Peng has been a key contributor in refining the framework and control plane protocol aspects.

19. Acknowledgments

This protocol is the outcome of work by many engineers, not just the authors of this document. In alphabetical order, the contributors to the project are: Barbara Orlandi, Bongho Kim,David Lopez-Perez, Doru Calin, Jonathan Ling, Lohith Nayak, Michael Scharf.

20. IANA Considerations

This draft makes no requests of IANA

21. References

21.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

21.2. Informative References

[ANDSF] "TS 24.312 version 15.0, 21 June 2018: 3GPP Specification on Access Network Discovery and Selection Function (ANDSF) Management Object (MO)", http://www.3gpp.org/ftp//Specs/archive/24_series/24.312/24312-f00.zip", <TS24.312>.

[E212] "ITU-T E.212: The international identification plan for public networks and subscriptions, https://www.itu.int/rec/T-REC-E.212-201609-I/en", <ITU-T E.212>.

[ETSIMAMS] "Multi-access Edge Computing (MEC); Phase 2: Use Cases and Requirements, https://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/02.01.01_60/gs_MEC002v020101p.pdf", <ETSI GS MEC 002>.

[ETSIMEC] "Multi-access Edge Computing (MEC), ETSI", <https://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing>.

[ETSIRNIS] "Mobile Edge Computing (MEC) Radio Network Information API", <ETSI GS MEC 012>.

[GPPATSSS] "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Study on Access Traffic Steering, Switching and Splitting support in the 5G system architecture (Release 16), https://www.3gpp.org, work in progress", <TR23.793>.

[I-D.deconinck-multipath-quic] Coninck, Q. and O. Bonaventure, "Multipath Extension for QUIC", draft-deconinck-multipath-quic-00 (work in progress), October 2017.

[I-D.ietf-tcpm-converters] Bonaventure, O., Boucadair, M., Gundavelli, S., Seo, S., and B. Hesmans, "0-RTT TCP Convert Protocol", draft-ietf-tcpm-converters-06 (work in progress), March 2019.

[I-D.zhu-intarea-gma] Zhu, J. and S. Kanugovi, "Generic Multi-Access (GMA) Convergence Encapsulation Protocols", draft-zhu-intarea-gma-02 (work in progress), March 2019.

[I-D.zhu-intarea-mams-user-protocol] Zhu, J., Seo, S., Kanugovi, S., and S. Peng, "User-Plane Protocols for Multiple Access Management Service", draft-zhu-intarea-mams-user-protocol-07 (work in progress), April 2019.

[IEEE] "IEEE Standard for Information technology: Telecommunications and information exchange between systems Local and metropolitan area networks:Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.", <IEEE 802.11-2016>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <https://www.rfc-editor.org/info/rfc2784>.

[RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC 2890, DOI 10.17487/RFC2890, September 2000, <https://www.rfc-editor.org/info/rfc2890>.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <https://www.rfc-editor.org/info/rfc3948>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <https://www.rfc-editor.org/info/rfc6824>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[SCFTECH5G] "Small Cell Forum, https://scf.io/", <Small Cell Forum>.

[WBAUnl5G] "Wireless Broadband Alliance Project - Unlicensed Integration with 5G Networks, https://www.wballiance.com/resource/ unlicensed-integration-with-5g-networks/.", <Unlicensed Integration with 5G Networks>.

Appendix A. MAMS Control Plane Optimization over Secure Connections

If the connection between CCM and NCM over which the MAMS control plane messages are transported is assumed to be secure, UDP is used as the transport for management & control messages between NCM and UCM (see Figure 23).

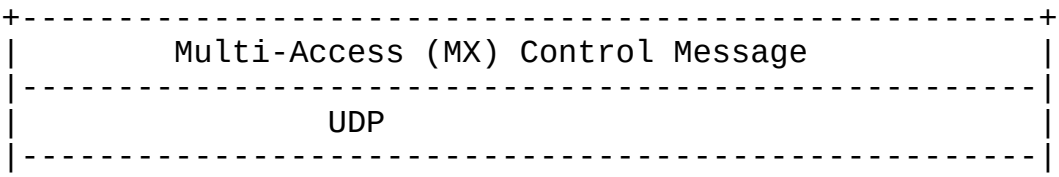

Figure 23: UDP-based MAMS Control plane Protocol Stack

Appendix B. MAMS Application Interface

B.1. Overall Design

CCM hosts an HTTPS server for applications to communicate and request services. It is assumed in this draft that all CCM and the communicating Application instances are hosted in a single administrative domain from security point of view.

The content of messages is defined in "Java Script Object Notation" (JSON) format. They offer RESTful APIs for communication.

The exact mechanism regarding how Application knows about the end point of CCM is not covered as part of this document. It may be provided as part of the Application Settings.

B.2. Notation

The documentation of APIs are provided in OpenAPI format using swagger v2.0 (TBD - Add section in appendix)

B.3. Error Indication

For every API, there could be an error response in case the objective of API could not be met as defined in [RFC2616].

B.4. CCM APIs

The following sections describe the APIs exposed by CCM to the applications

B.4.1. Get Capabilities

The CCM provides a HTTPS GET interface as "/ccm/v1.0/capabilities" for the Application to query about the capabilities supported by the CCM instance.

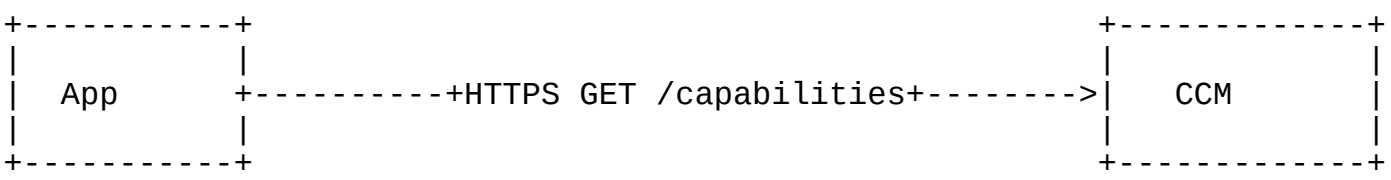

Figure 24: CCM API - GET Procedures

The CCM shall provide information of its capabilities as follows:

- Supported Features: One of more of as defined in MX Feature Activation List parameter of MX Capability REQ.
- Supported Connections: Supported connection types and connection IDs
- Supported MX Adaptation Sublayers: List of MX adaptation sublayer protocols supported by the N-MADP instance alongwith the connection type where these are supported and their respective parameters.
- Supported MX Convergence Sublayers: List of supported MX Convergence Sublayer protocols alongwith the parameters associated with respective convergence technique.

B.4.2. Post App Requirements

The CCM provides a HTTPS POST interface as "/ccm/v1.0/ app_requirements" for the Application to post the needs of the application data streams to the CCM instance.

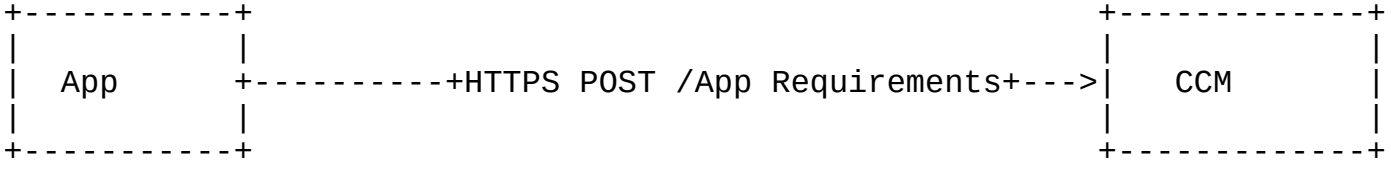

Figure 25: CCM API - POST Procedures

The CCM shall provide for the application to post the following requirements of its different data streams:

- Number of data stream types For each data stream type, the following link feature preferences,
- Protocol Type: Transport layer protocol associated with the application data stream packets.
- Port Range: Supported connection types and connection IDs.

- Traffic QoS: Quality of service parameters as follows
  - Bandwidth
  - Latency
  - Jitter

B.4.3. Get Predictive Link Parameters

The CCM provides a HTTPS GET interface as "/ccm/v1.0/predictive_link_params" for the Application to get the predicted link parameters from the CCM instance.

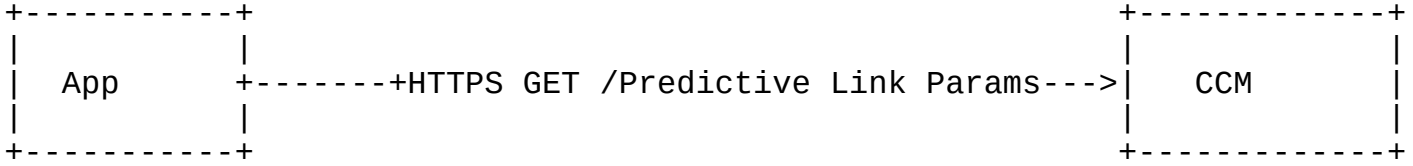

Figure 26: CCM API - GET Predictive Link Parameters Procedures

CCM requests the NCM for link parameters via the MAMS Network Analytics Request Procedure (Section 8.12) and includes the information in response to the API invocation.

- Number of Delivery Connections For Each Delivery Connection,
  - Access Link Identifier
    - Connection Type
    - Connection ID
  - Link Quality Indicator
    - Bandwidth
      - Predicted Value (in Mbps)
      - - Likelihoood (in Percentage)
      - Prediction Validity (Validity Time in s)
    - Jitter
      - Predicted Value (in s)
      - - Likelihoood (in Percentage)
      - Prediction Validity (Validity Time in s)
    - Latency
      - Predicted Value (in s)

- - Likelihoood (in Percentage)
- Prediction Validity (Validity Time in s)

+ Signal Quality

- if Delivery Connection Type is LTE, LTE_RSRP Predicted Value (in dBm)
- if Delivery Connection Type is LTE, LTE_RSRQ Predicted Value (in dBm)
- if Delivery Connection Type is NR, NR_RSRP Predicted Value (in dBm)
- if Delivery Connection Type is NR, NR_RSRQ Predicted Value (in dBm)
- if Delivery Connection Type is WiFi, WLAN_RSSI Predicted Value (in dBm)
- - Likelihoood (in Percentage)
- Prediction Validity (Validity Time in s)

Appendix C. JSON Specification for MAMS Control Plane

MAMS Control plane messages are exchanged between the CCM and the NCM. This section, specifies the format and content of messages in "Java Script Object Notation" (JSON) format.

C.1. Protocol Specification: General Processing

C.1.1. Notation

This document uses JSONString, JSONNumber, and JSONBool to indicate the JSON string, number, and boolean types, respectively. The type JSONValue indicates a JSON value, as specified in Section 3 of [RFC7159].

This document uses an adaptation of the C-style struct notation to define JSON objects. A JSON object consists of name/value pairs. This document refers to each pair as a field. In some context, this document also refers to a field as an attribute. The name of a field/attribute may be referred to as the key. An optional field is enclosed by [ ]. In the definitions, the JSON names of the fields are case sensitive. An array is indicated by two numbers in angle brackets, <m..n>, where m indicates the minimal number of values and n is the maximum. When this document uses * for n, it means no upper bound.

For example, the definition below defines a new type Type4, with three fields named "name1", "name2", and "name3", respectively. The field named "name3" is optional, and the field named "name2" is an array of at least one value.

object { Type1 name1; Type2 name2<1..*>; [Type3 name3;] } Type4;

This document uses subtyping to denote that one type is derived from another type. The example below denotes that TypeDerived is derived from TypeBase. TypeDerived includes all fields defined in TypeBase. If TypeBase does not have a field named "name1", TypeDerived will have a new field named "name1". If TypeBase already has a field named "name1" but with a different type, TypeDerived will have a field named "name1" with the type defined in TypeDerived (i.e., Type1 in the example).

object { Type1 name1; } TypeDerived : TypeBase;

Note that, despite the notation, no standard, machine-readable interface definition or schema is provided in this document. Extension documents may describe these as necessary.

C.1.2. Discovery Procedure

C.1.2.1. MX Discovery Message

This message is the first message sent by CCM to discover the presence of NCM in the network. It contains only the base information as described in Appendix C.2.1 with message_type set as mx_discover.

Following is the representation of the message:

object {

[JSONString MCC_MNC_Tuple;]

} MXDiscover : MXBase;

C.1.3. System Information Procedure

C.1.3.1. MX System Information Message

This message is sent by NCM to CCM to inform the endpoints that NCM supports for MAMS functionality. In addition to base information (Appendix C.2.1) it contains following information:

a) NCM Connections (described in Appendix C.2.3)

Following is the representation of the message:

object {

```
NCMConnections ncm_connections;

} MXSystemInfo : MXBase;
```

C.1.4. Capability Exchange Procedure

C.1.4.1. MX Capability Request

This message is sent by CCM to NCM to indicate the capabilities of the CCM instance available to the NCM indicated in System Info message earlier. In addition to base information (Appendix C.2.1) it contains following information:

(a) Features Activation Status: Described in Appendix C.2.5
(b) Number of anchor connections: Number of anchor connection (towards core) supported by the NCM.
(c) Anchor Connections: Described in sec Appendix C.2.6
(d) Number of Delivery connections: Number of delivery connection (towards access) supported by the NCM.
(e) Delivery Connections: Described in Appendix C.2.7
(f) Convergence Methods: Described in Appendix C.2.9
(g) Adaptation Methods: Described in Appendix C.2.10

Following is the representation of the message:

```
object {
FeaturesActive feature_active;
JSONNumber num_anchor_connections;
AnchorConnections anchor_connections;
JSONNumber num_delivery_connections;
DeliveryConnections delivery_connections;
ConvergenceMethods convergence_methods;
AdaptationMethods adaptation_methods
} MXCapabilityReq : MXBase;
```

C.1.4.2. MX Capability Response

This message is sent by NCM to CCM to indicate the capabilities of the NCM instance and unique session identifier for CCM. In addition to base information (Appendix C.2.1) it contains following information:

(a) Features Activation Status: Described in Appendix C.2.5
(b) Number of anchor connections: Number of anchor connection (towards core) supported by the NCM.
(c) Anchor Connections: Described in Appendix C.2.6
(d) Number of Delivery connections: Number of delivery connection (towards access) supported by the NCM.

(e) Delivery Connections: Described in Appendix C.2.7
(f) Convergence Methods: Described in Appendix C.2.9
(g) Adaptation Methods: Described in Appendix C.2.10
(h) Unique Session Id: This uniquely identifies the session between CCM and NCM in a network. Described in Appendix C.2.2

Following is the representation of the message:

```
object {
FeaturesActive feature_active;
JSONNumber num_anchor_connections;
AnchorConnections anchor_connections;
JSONNumber num_delivery_connections;
DeliveryConnections delivery_connections;
ConvergenceMethods convergence_methods;
AdaptationMethods adaptation_methods
UniqueSessionId unique_session_id;
} MXCapabilityRsq : MXBase;
```

C.1.4.3. MX Capability Acknowledge

This message is sent by CCM to NCM to indicate acceptance of capabilities advertised by NCM in earlier MX Capability Response message. In addition to base information (Appendix C.2.1) it contains following information:

(a) Unique Session Id: Same identifier as provided in MX Capability RSP. Described in Appendix C.2.2.
(b) Capability Acknowledgement: Either Accept or Reject of the capabilities sent by CCM. Can take either "MX_ACCEPT" or "MX_REJECT" as acceptable values.

Following is the representation of the message:

```
object {
UniqueSessionId unique_session_id;
JSONString capability_ack;
} MXCapabilityAck : MXBase;
```

C.1.5. User Plane Configuration Procedure

C.1.5.1. MX User Plane Configuration Request

This message is sent by NCM to CCM to configure the user plane for MAMS. In addition to base information (Appendix C.2.1) it contains following information:

(a) Number of Anchor Connection: Number of anchor connections supported by NCM.
(b) Setup of Anchor Connections: Described in Appendix C.2.11.

Following is the representation of the message:

```
object {
JSONNumber num_anchor_connections;
SetupAnchorConns anchor_connections;
} MXUPSetupConfigReq : MXBase;
```

C.1.5.2. MX User Plane Configuration Confirmation

This message is the confirmation of user plane setup message sent from CCM after successfully configuring the user plane at user equipment. This message contains following information:

(a) Unique Session Id: Same identifier as provided in MX Capability RSP. Described in Appendix C.2.2.
(b) MX Probe Parameters (included if probing is supported)

    (1) Probe Port: UDP port for accepting probe message.
    (2) Anchor connection Id: Identifier of the anchor connection to be used for probe function, provided in user plane setup request.
    (3) MX Configuration Id: For the given anchor connection, which configuration id is to be used for probe, this is present only if provided in the user plane setup request.
(c) For each delivery connection following is required:

    (1) Connection ID: Delivery connection id supported by UE.
    (2) Client Adaptation Layer Parameters: If UDP adaptation layer is in use then the UDP port to be used at C-MADP side.

Following is the representation of the message:

```
object {
UniqueSessionId unique_session_id;
[ProbeParam probe_param;]
JSONNumber num_delivery_conn;
ClientParam client_params <1...*>;
} MXUPSetupConfigCnf : MXBase;
```

Where ProbeParam is defined as following:

```
object {
JSONNumber probe_port;
JSONNumber anchor_conn_id;
```

```
   [JSONNumber mx_configuration_id;]
   } ProbeParam;
```

Where ClientParam is defined as following:

```
   object {
   JSONNumber connection_id;
   [AdaptationParam adapt_param;]
   } ClientParam;
```

Where AdaptationParam is defined as following:

```
   object {
   JSONNumber udp_adapt_port;
   } AdaptationParam;
```

C.1.6. Reconfiguration Procedure

C.1.6.1. Reconfiguration Request

This message is sent by CCM to NCM in case of reconfiguration of any the connections from user equipment's side. In addition to base information (Appendix C.2.1) it contains following information:

(a) Unique Session Id: Identifier for CCM-NCM association Appendix C.2.2.
(b) Reconfiguration Action: Type of reconfiguration action can be one of "setup", "release" or "modify".
(c) Connection Id: Connection Id for which the reconfiguration is taking place.
(d) IP address: IP address in case of setup and modify type of reconfiguration.
(e) SSID: If the connection type is WiFi, in that case the SSID the UE has attached to is contained in this parameter.
(f) MTU of connection: MTU of the delivery path that is calculated at the UE for use by NCM to configure fragmentation and concatenation procedures at N-MADP.
(g) Connection Status: This parameter informs if the connection is currently "disabled", "enabled" or "connected". Default: "connected".
(h) Delivery Node Id: Identity of the node to which the client is attached. ECGI in case of LTE and WiFi AP Id or MAC address in case of WiFi.

Following is the representation of the message:

```
   object {
   UniqueSessionId unique_session_id;
```

```
      JSONString reconf_action;
      JSONNumber connection_id;
      JSONString ip_address;
      JSONString ssid;
      JSONNumber mtu_size;
      JSONString connection_status;
      [JSONSring delivery_node_id;]
      } MXReconfReq : MXBase;
```

C.1.6.2. Reconfiguration Response

This message is sent by NCM to CCM as a confirmation towards reconfiguration requirement after taking the reconfiguration into use and contains only the base information (as defined in Appendix C.2.1).

Following is the representation of the message:]

```
      object {
      } MXReconfRsp : MXBase;
```

C.1.7. Path Estimation Procedure

C.1.7.1. Path Estimation Request

This message is sent by NCM towards CCM to configure the CCM to send path estimation reports. In addition to base information (Appendix C.2.1) it contains following information:

(a) Connection Id: Id of the connection for which the path estimation report is required.
(b) Init Probe Test Duration: Duration of initial probe test in milliseconds. [TBD: Range of values]
(c) Init Probe Test Rate: Initial testing rate in Mega Bits per Second. [TBD: Range of values]
(d) Init Probe Size: Size of each packet for initial probe in Bytes. [TBD: Range of values]
(e) Init Probe Ack: If an acknowledgement for probe is required. [Possible values: "yes", "no"]
(f) Active Probe Frequency: Frequency in milliseconds at which the active probes shall be sent. [TBD: Range of values]
(g) Active Probe Size: Size of the active probe in Bytes. [TBD: Range of values]
(h) Active Probe Duration: Duration in seconds for which the active probe shall be performed. [TBD. Range of values]
(i) Active Probe Ack. If an acknowledgement for probe is required. [Possible values: "yes", "no"]

Following is the representation of the message:

```
object {
JSONNumber connection_id;
JSONNumber init_probe_test_duration_ms;
JSONNumber init_probe_test_rate_Mbps;
JSONNumber init_probe_size_bytes;
JSONString init_probe_ack_req;
JSONNumber active_probe_freq_ms;
JSONNumber active_probe_size_bytes;
JSONNumber active_probe_duration_sec;
JSONString active_probe_ack_req;
} MXPathEstReq : MXBase;
```

C.1.7.2. Path Estimation Report

This message is sent by CCM to NCM as report to the probe estimation configured by NCM. In addition to base information (Appendix C.2.1) it contains following information:

(a) Unique Session Id: Same identifier as provided in MX Capability RSP. Described in Appendix C.2.2.
(b) Connection Id: Id of the connection for which the path estimation report is required.
(c) Init Probe Results: Defined in section Appendix C.2.12.
(d) Active Probe Results: Defined in section Appendix C.2.13.

Following is the representation of the message:

```
object {
JSONNumber connection_id;
UniqueSessionId unique_session_id;
[InitProbeResults init_probe_results;]
[ActiveProbeResults active_probe_results;]
} MXPathEstResults : MXBase;
```

C.1.8. Traffic Steering Procedure

C.1.8.1. Traffic Steering Request

This message is sent by NCM to CCM for enabling traffic steering at delivery side in uplink and downlink configuration. In addition to base information (Appendix C.2.1) it contains following information:

(a) Connection id: Anchor connection number for which the traffic steering is getting defined.
(b) MX Configuration Id: MX configuration for which the traffic steering is getting defined.

(c) Downlink Delivery: Defined in Appendix C.2.14.
(d) Default UL Delivery: The default delivery connection for uplink. All traffic should be delivered on this connection in uplink direction and the TFT filter should be applied only for the traffic mentioned in Uplink Delivery.
(e) Uplink Delivery: Defined in Appendix C.2.15.
(f) Features Activated: Defined in Appendix C.2.5.

Following is the representation of the message:

```
object {
JSONNumber connection_id;
[JSONNumber mx_configuration_id;]
DLDelivery downlink_delivery;
JSONNumber default_uplink_delivery;
ULDelivery uplink_delivery;
FeaturesActive feature_activation;
} MXTraffiSteeringReq : MXBase;
```

C.1.8.2. Traffic Steering Response

This message is response to Traffic Steering request from CCM to NCM. In addition to base information (Appendix C.2.1) it contains following information:

(a) Unique Session Id: Same identifier as provided in MX Capability RSP. Described in Appendix C.2.2.
(b) Features Activated: Defined in Appendix C.2.5.

Following is the representation of the message:

```
object {
UniqueSessionId unique_session_id;
FeaturesActive feature_activation;
} MXTraffiSteeringResp : MXBase;
```

C.1.9. MAMS Application MADP Association

C.1.9.1. MAMS Application MADP Association Request

This message is sent by CCM to NCM to select MADP instances provided earlier in User Plane Setup Request based on requirement of the applications.

In addition to the base information (Appendix C.2.1) it contains following:

(a) Unique Session Id: This uniquely identifies the session between CCM and NCM in a network. Described in Appendix C.2.2, and a list of following MX Application MADP Associations

  (a) Connection id: Defines the anchor connection number of the MADP instance
  (b) MX Configuration Id: identify the MX configuration of the MADP instance
  (c) Traffic Template Uplink: Traffic template as defined in 5.16 to be used in uplink direction.
  (d) Traffic Template Downlink: Traffic template as defined in 5.16 to be used in downlink direction.

Following is the representation of the message:

```
object {
UniqueSessionId unique_session_id;
MXAppMADPAssoc app_madp_assoc_list <1..*>;
} MXAppMADPAssocReq : MXBase;
```

Where each measurement MXAppMADPAssoc is represented by following:

```
object {
JSONNumber connection_id;
JSONNumber mx_configuration_id
TrafficFlowTemplate tft_ul_list <1..*>;
TrafficFlowTemplate tft_dl_list <1..*>;
} MXAppMADPAssoc;
```

C.1.9.2. MAMS Application MADP Association Response

This message is sent by NCM to CCM to confirm the selected MADP instances provided in request message by CCM.

In addition to the base information (Appendix C.2.1), it contains information if the request has been successful.

Following is the representation of the message:

```
object {
JSONBool is_success;
} MXAppMADPAssocResp : MXBase;
```

C.1.10. SSID Indication

This message is sent by NCM to CCM to indicate the list of allowed SSID which are supported by MAMS entity at the network side. It contains the list of SSIDs.

Each SSID consists of the type of SSID (which can be one of the "SSID", "BSSID" or "HESSID" and the SSID itself.

Following is the representation of the message:

```
object {
SSID ssid_list<1..*>;
} MXSSIDIndication : MXBase;
```

Where each SSID is defined as following:

```
object {
JSONString ssid_type;
JSONString ssid;
} SSID;
```

C.1.11. Measurements

C.1.11.1. Measurement Configuration

This message is sent from NCM to CCM to configure the period measurement reporting at CCM. The message contains a list of measurement configuration with each element containing following information:

(a) Connection Id: Connection id of the delivery connection for which the reporting is being configured.
(b) Connection Type: Connection Type for which the reporting is being configured, can be "lte", "wifi", "5g-nr" etc.
(c) Measurement Report Configuration: Actual report configuration based on the connection type, as defined in Appendix C.2.17

Following is the representation of the message:

```
object {
MeasReportConf measurement_configuration <1..*>;
} MXMeasReportConf : MXBase;
```

Where each measurement MeasReportConf is represented by following:

```
object {
JSONNumber connection_id;
JSONString connection_type;
MeasReportConfs meas_rep_conf <1..*>;
} MeasReportConf;
```

C.1.11.2. Measurement Report

This message is periodically sent by CCM to NCM after measurement configuration. In addition to the base information it contains following information:

(a) Unique Session Id: Same identifier as provided in MX Capability RSP. Described in Appendix C.2.2.
(b) Measurement report for each delivery connection is measured by the client device as defined in Appendix C.2.18.

Following is the representation of the message:

```
object {
UniqueSessionId unique_session_id;
MXMeasRep measurment_reports <1..*>;
} MXMeasurementReport : MXBase;
```

C.1.12. Keep Alive

C.1.12.1. Keep Alive Request

A Keep Alive Request message can be sent from either NCM or CCM on expiry of MAMS_KEEP_ALIVE timer or a handover event. This request shall be responded by the peer with Keep Alive Response. In case of no response from peer the MAMS connection shall be assumed to be broken and new connection shall be established again by CCM by sending MX Discover messages.

In addition to the base information it cantains following information:

(a) Keep Alive Reason: Reason for sending this message, can be "Timeout" or "Handover".
(b) Unique Session Id: Identifier for CCM-NCM association Appendix C.2.2.
(c) Connection Id: Connection id for which handover is detected, in case the reason is "Handover".
(d) Delivery Node Id: The target delivery node id (ECGI or WiFi Access Point Id/MAC) to which the handover is executed.

Following is the representation of the message:

```
object {
JSONString keep_alive_reason;
UniqueSessionId unique_session_id;
JSONNumber connection_id;
JSONString delivery_node_id;
```

```
} MXKeepAliveReq : MXBase;
```

C.1.12.2. Keep Alive Response

On receiving Keep Alive Request from peer, NCM/CCM shall immediately respond with a Keep Alive Response message on the same delivery path from where the request arrived. In addition to base information it contains the unique session identifier for the CCM-NCM association (defined in Appendix C.2.2)

Following is the representation of the message:

```
object {
UniqueSessionId unique_session_id;
} MXKeepAliveResp : MXBase;
```

C.1.13. Session Termination Procedure

C.1.13.1. Session Terminate Request

In the event where NCM or CCM can no longer handle MAMS for any reason then it can send MX session termination request to the peer. In addition to base information it contains Unique Session Id and reason for termination, this can be "MX_NORMAL_RELEASE", "MX_NO_RESPONSE" or "INTERNAL_ERROR".

Following is the representation of the message:

```
object {
UniqueSessionId unique_session_id;
JSONString reason;
} MXSessionTerminationReq : MXBase;
```

C.1.13.2. Session Terminate Response

On reception of MX session termination request from peer, NCM/CCM shall respond with MX Session Termination Response on the same delivery path where the request arrived and clean the MAMS related resources and settings. CCM shall re-initiate a new session with MX Discover messages again.

Following is the representation of the message:

```
object {
UniqueSessionId unique_session_id;
} MXSessionTerminationResp : MXBase;
```

C.1.14. Network Analytics

C.1.14.1. MX Network Analytics Request

This message is sent by CCM to NCM to request for parameters like bandwidth, jitter, latency and signal quality predicted by network analytics function. In addition to the base information it contains following parameter:

(a) Unique Session Id: Same identifier as provided in MX Capability RSP. Described in Appendix C.2.2.
(b) Parameter List: List of parameters which the CCM is interested in namely one or more of, "bandwidth", "jitter", "latency" and "signal_quality".

Following is the representation of the message:

```
object {
UniqueSessionId unique_session_id;
JSONString params<1..*>;
} MXNetAnalyticsReq : MXBase;
```

Where the params object can take one or more of the following values:

```
"bandwidth"
"jitter"
"latency"
"signal_quality"
```

C.1.14.2. MX Network Analytics Response

This message is sent by NCM to CCM in response to the analytics request. For each delivery connection that the client has NCM reports the requested parameter predictions and their respective likelihood (between 1-100).

In addition to the base information it contains following parameters:

(a) Number of delivery connections: The number of delivery connections configured for the client currently.
(b) For each delivery connection following is provided:

    (a) Connection Id: Connection ID of the delivery connection for which the parameters are being predicted.
    (b) Connection Type: Type of connection, can be "Wi-Fi, "5G NR", "Multi-Fire" and "LTE".
    (c) List of Parameters for which Prediction is requested, where each of the predicted parameters consists of following:

(a) Parameter name: Name of the parameter being predicted. Can be one of "bandwidth", "jitter", "latency", "signal_quality".
(b) Additional Parameter: If Parameter name is "signal_quality", then this qualifies the quality parameter like "lte_rsrp", "lte_rsrq", "nr_rsrp", "nr_rsrq", or "wifi_rssi".
(c) Predicted value: Provides the predicted value of the parameter and, if applicable, the additional parameter.
(d) Likelihood: Provides a stochastic likelihood of about the predicted value.
(e) Validity Time: the time horizon until which the predictions are valid.

Following is the representation of the message:

```
object {
MXAnalyticsList param_list<1..*>>;
} MXNetAnalyticsResp : MXBase;
```

Where MXAnalyticsList is defined as following::

```
object{
JSONNumber connection_id;
JSONString connection_type;
ParamPredictions predictions <1..*>;
} MXAnalyticsList;
```

Where each ParamPredictions is defined as:

```
object{
JSONString param_name;
[JSONString additional_param;]
JSONNumber prediction;
JSONNumber likelihood;
JSONNumber validity_time;
} ParamPredictions;
```

C.2. Protocol Specification: Data Types

C.2.1. MXBase

This is the base information that every message between CCM and NCM exchanges shall have as mandatory information. It contains following information:

(a) Version: Version of MAMS in used

(b) Message Type: Message type being sent with following as valid values:

(a) "mx_discover"
(b) "mx_system_info"
(c) "mx_capability_req"
(d) "mx_capability_resp"
(e) mx_capability_ack"
(f) "mx_up_setup_conf_req"
(g) "mx_up_setup_cnf"
(h) "mx_reconf_req"
(i) "mx_reconf_rsp"
(j) "mx_path_est_req"
(k) "mx_path_est_results"
(l) "mx_traffic_steering_req"
(m) "mx_traffic_steering_rsp"
(n) "mx_ssid_indication"
(o) "mx_keep_alive_req"
(p) "mx_keep_alive_rsp"
(q) "mx_measurement_conf"
(r) "mx_measurement_report"
(s) "mx_session_termination_req"
(t) "mx_session_termination_resp"
(u) "mx_app_madp_assoc_req"
(v) "mx_app_madp_assoc_resp"
(w) "mx_network_analytics_req"
(x) "mx_network_analytics_resp"

(c) Sequence Number: Sequence number to uniquely identify a transaction of message exchange, e.g. MX Capability REQ/RSP/ ACK.

Following is the representation of this data type:

```
object {
JSONString version;
JSONString message_type;
JSONNumber sequence_num;
} MXBase;
```

C.2.2. Unique Session Id

This data type defines the unique session id between a CCM and NCM entity, it contains a NCM id which is unique in the network and a session id allocated by NCM for that session. On reception, of discovery message if the session is existing then the old session id is returned in System Info message otherwise NCM allocates a new session id to the CCM and sends in response with System Info message.

Following is the representation of this data type:

```
object {
JSONNumber ncm_id;
JSONNumber session_id;
} UniqueSessionId;
```

C.2.3. NCM Connections

This data type defines the connection available at NCM for MAMS connectivity towards the User Equipment. It contains a list of NCM connections available where each connection has following information:

(a) Connection Information: As defined in Appendix C.2.4
(b) NCM End Point information: This contains IP Address and Port exposed by NCM end point for CCM.

Following is the representation of this data type:

```
object {
NCMConnection items<1..*>;
} NCMConnections;
```

where NCMConnection is defined as:

```
object {
NCMEndPoint ncm_end_point;
} NCMConnection : ConnectionInfo;
```

where NCMEndPoint is defined as:

```
object {
JSONString ip_address;
JSONNumber port;
} NCMEndPoint;
```

C.2.4. Connection Information

This data type provides the mapping of connection Id and connection type. It contains following information:

(a) Connection Id: Number indicating the connection can be 0,1,2 and 3.
(b) Connection type: Type of connect can be "Wi-Fi, "5G NR", "Multi-Fire" and "LTE".

The two are considered a mapping like 0-"Wi-Fi", 1-"5G NR", 2-"Multi-Fire" and 3-"LTE".

Following is the representation of this data type:

```
object {
JSONNumber connection_id;
JSONString connection_type;
} ConnectionInfo;
```

C.2.5. Features Activation Status

This data type provides the list of all features with their activation status. Each feature status contains following:

(a) Feature Name: Name of the feature can be one of the following:

  (a) "lossless_switching"
  (b) "fragmentation"
  (c) "concatenation"
  (d) "uplink_aggregation"
  (e) "downlink_aggregation"
  (f) "measurement"

(b) Active status: Activation status of the feature, "true" means feature is active, "false" means feature is inactive.

Following is the representation of this data type:

```
object {
FeatureInfo items<1..*>;
} FeaturesActive;
```

where FeatureInfo is defined as:

```
object {
JSONString feature_name;
JSONBool active;
} FeatureInfo;
```

C.2.6. Anchor Connections

This data type contains the list of Connection Information (Appendix C.2.4) that are supported at anchor (core) side.

Following is the representation of this data type:

```
object {
ConnectionInfo items<1..*>;
```

```
} AnchorConnections;
```

C.2.7. Delivery Connections

This data type contains the list of Connection Information (Appendix C.2.4) that are supported at delivery (access) side.

Following is the representation of this data type:

```
object {
ConnectionInfo items<1..*>;
} DeliveryConnections;
```

C.2.8. Method Support

This data type provides the support for a particular convergence or adaptation method. It consists of following:

(a) Method: Name of the method.
(b) Supported: Whether the method named above is supported or not. Possible values are "true" and "false".

Following is the representation of this data type:

```
object {
JSONString method;
JSONBool supported;
} MethodSupport;
```

C.2.9. Convergence Methods

This data type contains the list of all convergence methods and their support status. Convergence Methods possible are:

```
"GMA"
"MPTCP_Proxy"
"GRE_Aggregation_Proxy"
"MPQUIC"
```

Following is the representation of this data type:

```
object {
MethodSupport items<1..*>;
} ConvergenceMethods;
```

C.2.10. Adaptation Methods

This data type contains the list of all convergence methods and their support status. Converge Methods possible are:

"UDP_without_DTLS"
"UDP_with_DTLS"
"IPSec"
"Client_NAT"

Following is the representation of this data type:

```
object {
MethodSupport items<1..*>;
} AdaptationMethods;
```

C.2.11. Setup of Anchor Connections

This data type defines the setup configuration for each of the anchor connection that is required at the user equipment side. It contains following information in addition to the connection id and type of the anchor connection:

(a) Number of Active MX configurations: If more than one active configurations are present for this anchor then this identifies the number of such connections
(b) For each active configuration following convergence parameters are provided:

    (a) MX Configuration Identifier: This identifier is present in case there are multiple active configuration and identifies the configuration for this MADP instance id.
    (b) Convergence Method: Converge method selected, has to be one of the supported convergence method as listed in section Appendix C.2.9.
    (c) Convergence Method Parameters: Described in section Appendix C.2.11.1
    (d) Number of Delivery Connections: Number of delivery connections (access side) that are supported for this anchor connection.
    (e) Setup Delivery Connections: Described in section Appendix C.2.11.2.

Following is the representation of this data type:

```
object {
SetupAnchorConn items<1..*>;
} SetupAnchorConns;
```

Where each Anchor connection configuration is defined as following:

```
object {
[JSONNumber num_active_mx_conf;]
ConvergenceConfig convergence_config
} SetupAnchorConn : ConnectionInfo;
```

where each Convergence configuration is defined as following:

```
object {
[JSONNumber mx_configuration_id;]
JSONString convergence_method;
ConvergenceMethodParam convergence_method_params;
JSONNumber num_delivery_connections;
SetupDeliveryConns delivery_connections;
} ConvergenceConfig;
```

C.2.11.1. Convergence Method Parameters

This data type defines the parameters used for convergence method and contains following:

(a) Proxy IP: IP Address of proxy that is provided by Convergence Method selected.
(b) Proxy Port: Port of the proxy that is provided by Convergence Method selected.

Following in the representation of this data type:

```
object {
JSONString proxy_ip;
JSONString proxy_port;
JSONString client_key;
} ConvergenceMethodParam;
```

C.2.11.2. Setup Delivery Connections

This is the list of delivery connections and their parameters to be configured at the user equipment. Each delivery connection defined by its connection information (Appendix C.2.4) contains optionally following:

(a) Adaptation Method: Selected adaptation method name, this shall be one of the names as listed in Appendix C.2.10.
(b) Adaptation Method Parameters: Depending on the adaptation method one or more of the following parameters shall be provided.

    (a) Tunnel IP address

(b) Tunnel Port number
(c) Shared Secret
(d) MX header optimization: If the adaptation method is UDP_and convergence is GMA then this flag represents if the checksum field and the length field in the IP header of a MX PDU should be recalculated or not by the MX convergence sublayer. The possible values are "true" and "false". If it is "true", both fields remain unchanged; otherwise, both fields should be recalculated. If this field is not present then the default of "false" should be considered.

Following in the representation of this data type:

```
object {
SetupDeliveryConn items<1..*>;
} SetupDeliveryConns;
```

where each Setup Delivery Connection consists of following:

```
object {
[JSONSting adaptation_method;]
[AdaptationMethodParam adaptation_method_param;]
} SetupDeliveryConn : ConnectionInfo;
```

where Adaptation Method Param is defined as:

```
object {
JSONString tunnel_ip_addr;
JSONString tunnel_end_port;
JSONString shared_secret;
[JSONBool mx_header_optimization;]
} AdaptationMethodParam;
```

C.2.12. Init Probe Results

This data type defines the results of init probe request made by NCM. It consists of following information:

(a) Lost Probes: Percentage or probes lost.
(b) Prode Delay: Average delay of probe message in microseconds.
(c) Probe Rate: Probe rate achieved in Mega Bits per second.

Following in the representation of this data type:

```
object {
JSONNumber lost_probes_percentage;
JSONNumber probe_rate_Mbps;
} InitProbeResults;
```

C.2.13. Active Probe Results

This data type defines the results of init probe request made by NCM. It consists of following information:

(a) Average Probe Throughput: Average active probe throughput achieved in Mega Bits per second.

Following in the representation of this data type:

```
object {
JSONNumber avg_tput_last_probe_duration_Mbps;
} ActiveProbeResults;
```

C.2.14. Downlink Delivery

This data type defines the list of connections which are enabled in delivery side to be used in downlink direction.

Following in the representation of this data type:

```
object {
JSONNumber connection_id <1..*>;
} DLDelivery;
```

C.2.15. Uplink Delivery

This data type defines the list of connections and parameters enabled for deliver side to be used in uplink direction.

The uplink delivery consists of multiple uplink delivery entities, where each entity consists of a traffic flow template (TFT) Appendix C.2.16 and list of connection ids in uplink where traffic qualifying for such traffic flow template can be redirected.

Following in the representation of this data type:

```
object {
ULDeliveryEntity ul_del <1..*>;
} ULDelivery;
```

Where each uplink delivery entity consists of following data type:

```
object {
TrafficFlowTemplate ul_tft <1..*>;
JSONNumber connection_id <1..*>;
} ULDeliveryEntity;
```

C.2.16. Traffic Flow Template

Traffic flow template follows in general guidelines specified in 3GPP TS 23.060.

Traffic flow template in MAMS consists of one or more of following:

(a) Remote Address and Mask: IP address and subnet for remote addresses represented in CIDR notation. Default: "0.0.0.0/0".
(b) Local Address and Mask: IP address and subnet for local addresses represented in CIDR notation. Default: "0.0.0.0/0"
(c) Protocol Type: IP protocol number of the payload being carried by IP packet. e.g. UDP, TCP etc. Default: 255.
(d) Local Port Range: Range of ports for local ports for which the flow template is applicable. Default: Start=0, End=65535.
(e) Remote Port Range: Range of ports for remote ports for which the flow template is applicable. Default: Start=0, End=65535.
(f) Traffic Class: Represented by Type of Service in IPv4 and Traffic Class in IPv6. Default: 255
(g) Flow Label: Flow label for IPv6, applicable only for IPv6 protocol type. Default: 0.

Following in the representation of this data type:

```
object {
JSONString remote_addr_mask;
JSONString local_addr_mask;
JSONNumber protocol_type;
PortRange local_port_range;
PortRange remote_port_range;
JSONNumber traffic_class;
JSONNumber flow_label;
} TrafficFlowTemplate;
```

Where the port range is defined as following:

```
object {
JSONNumber start;
JSONNumber end;
} PortRange;
```

C.2.17. Measurement Report Configuration

This data type defines the configuration done by NCM towards CCM for reporting measurement events.

(a) Measurement Report Parameter: Parameter which shall be measured and reported. This is dependent on the connection type:

(a) For connection type "wifi" allowed measurement parameters are "WLAN_RSSI", "WLAN_LOAD", "UL_TPUT", "DL_TPUT", "EST_UL_TPUT" and "EST_DL_TPUT".
(b) For connection type "lte" allowed measurement parameters are "LTE_RSRP", "LTE_RSRQ", "UL_TPUT" and "DL_TPUT".
(c) For connection type "5g-nr" allowed measurement parameters are "NR_RSRP", "NR_RSRQ", "UL_TPUT" and "DL_TPUT".

(b) Threshold: High and Low threshold for reporting.
(c) Period: Period for reporting in milliseconds.

Following is the representation of this data type:

```
object {
JSONString meas_rep_param;
Threshold meas_threshold;
JSONNumber meas_period;
} MeasReportConfs;
```

Where Threshold is defined as following:

```
object {
JSONNumber high;
JSONNumber low;
} Threshold;
```

C.2.18. Measurement Report

This data type defines the measurements reported by CCM for each access network measured. This type contains the connection information, delivery node id which identifies the cell (ECGI) or the WiFI Access Point Id or MAC address (or equivalent identifier in other technologies) and the actual measurement performed by CCM in the last measurement period.

Following is the representation of this data type:

```
object {
JSONNumber connection_id;
JSONString connection_type;
JSONString delivery_node_id;
Measurement measurements <1..*>;
}MXMeasRep;
```

Where Measurement is defined as the key value pair of measurement type and value. The exact type and value are defined on a per delivery type and defined in Appendix C.2.17.

```
object{
```

```
      JSONString measurement_type;
      JSONNumber measurement_value;
      } Measurement;
```

C.3. Schemas in JSON

C.3.1. MX Base Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {
        "message_type_def": {
    "enum": [
    "mx_discover",
                "mx_system_info",
                "mx_capability_req",
                "mx_capability_resp",
                "mx_capability_ack",
                "mx_up_setup_conf_req",
                "mx_up_setup_cnf",
                "mx_reconf_req",
                "mx_reconf_rsp",
                "mx_path_est_req",
                "mx_path_est_results",
                "mx_traffic_steering_req",
                "mx_traffic_steering_rsp",
                "mx_ssid_indication",
                "mx_keep_alive_req",
                "mx_keep_alive_rsp",
                "mx_measurement_conf",
                "mx_measurement_report",
                "mx_session_termination_req",
                "mx_session_termination_resp",
    "mx_app_madp_assoc_req",
                "mx_app_madp_assoc_resp",
                "mx_network_analytics_req",
                "mx_network_analytics_resp"
      ],
          "type": "string"
    },
    "sequence_num_def": {
      "minimum": 1,
      "type": "integer"
    },
    "version_def": {
      "type": "string"
    }
   },
  "id": "http://www.ietf.org/mams/mx_base_def.json"
}
```

C.3.2. MX Definitions

```
{
        "$schema": "http://json-schema.org/draft-04/schema#",
        "definitions": {
```

```
"adapt_method": {
        "enum": [
                "UDP_without_DTLS",
                "UDP_with_DTLS",
                "IPSec",
                "Client_NAT"
        ],
        "type": "string"
},
"conv_method": {
        "enum": [
                "GMA",
                "MPTCP_Proxy",
                "GRE_Aggregation_Proxy",
                "MPQUIC"
        ],
        "type": "string"
},
"supported": {
        "type": "boolean"
},
"active": {
        "type": "boolean"
},
"connection_id": {
        "type": "integer"
},
"feature_name": {
        "enum": [
                "lossless_switching",
                "fragmentation",
                "concatenation",
                "uplink_aggregation",
                "downlink_aggregation",
                "measurement"
        ],
        "type": "string"
},
"connection_type": {
        "enum": [
                "wi-fi",
                "5g-nr",
                "multi-fire",
                "lte"
        ],
        "type": "string"
},
"ip_address": {
```

```
                        "type": "string"
                },
                "port": {
                        "maximum": 65535,
                        "minimum": 1,
                        "type": "integer"
                },
                "adaptation_method": {
                        "allOf" : [
                                { "$ref": "#/definitions/adapt_method" },
                                { "$ref": "#/definitions/supported" }
                        ]
                },
                "connection": {
                        "allOf" : [
                                { "$ref": "#/definitions/connection_id" },
                                { "$ref": "#/definitions/connection_type" }
                        ]
                },
                "convergence_method": {
                        "allOf": [
                                { "$ref": "#/definitions/conv_method" },
                                { "$ref": "#/definitions/supported" }
                        ]
                },
                "feature_status": {
                        "allOf": [
                                { "$ref": "#/definitions/feature_name" },
                                { "$ref": "#/definitions/active" }
                        ]
                },
                "ncm_end_point": {
                        "allOf" : [
                                { "$ref" : "#/definitions/ip_address" },
                                { "$ref" : "#/definitions/port" }
                        ]
                },
                "capability_acknowledgement" : {
                        "enum" : [
                                "MX_ACCEPT",
                                "MX_REJECT"
                        ],
                        "type" : "string"
                },
                "threshold" : {
                        "high" : {
                                "type" : "integer"
                        },
```

```
                "low" : {
                        "type" : "integer"
                },
                "type" : "object"
        },
        "meas_report_param" : {
                "enum" : [
                                "WLAN_RSSI",
                                "WLAN_LOAD",
                                "LTE_RSRP",
                                "LTE_RSRQ",
                                "UL_TPUT",
                                "DL_TPUT",
                                "EST_UL_TPUT",
                                "EST_DL_TPUT",
                                "NR_RSRP",
                                "NR_RSRQ",
                ],
                "type" : "string"
        },
        "meas_report_conf" : {
                "meas_rep_param" : {
                        "$ref" : "#definitions/meas_report_param"
                },
                "meas_threshold" : {
                        "$ref" : "#definitions/threshold"
                },
                "meas_period_ms" : {
                        "type" : "integer"
                },
                "type" : "object"
        },
        "ssid_types" : {
                "enum" : [
                        "ssid",
                        "bssid",
                        "hessid"
                ],
                "type" : "string"
        },
        "ip_addr_mask" : {
                "type" : "string",
                "default" : "0.0.0.0/0"
        },
        "port_range" : {
                "start" : {
                        "type" : "integer",
                        "default" : 0
```

```
                        },
                        "end" : {
                                "type" : "integer",
                                "default" : 65535
                        }
                },
                "traffic_flow_template" : {
                        "remote_addr_mask" : { "$ref" : "#definitions/ip_add
r_mask" },
                        "local_addr_mask" : {  "$ref" : "#definitions/ip_add
r_mask" },
                        "protocol_type" : {
                                "type" : "integer",
                                "minimum" : 0,
                                "maximum" : 255
                        },
                        "local_port_range" : {  "$ref" : "#definitions/port_
range" },
                        "remote_port_range" : { "$ref" : "#definitions/port_
range" },
                        "traffic_class" : {
                                "type" : "integer",
                                "default" : 255
                        },
                        "flow_label" : {
                                "type" : "integer",
                                "default" : 0
                        }
                },
                "delivery_node_id" : {
                        "type" : "string"
                },
                "unique_session_id" : {
                        "type" : "object",
                        "ncm_id" : {
                          "type" : "integer"
                        },
                        "session_id" : {
                          "type" : "integer"
                        }
                },
                "keep_alive_reason" : {
                        "enum" : [
                                "Timeout",
                                "Handover"
                        ],
                        "type" : "string"
                },
                "connection_status" : {
                        "enum" : [
                                "disabled",
                                "enabled",
```

```
                                "connected"
                        ],
                        "type" : "string",
                        "default" : "connected"
                },
                "adaptation_param" : {
                   "udp_adapt_port" : {
                      "type" : "integer"
                   }
                },
                "probe_param" : {
                    "probe_port" : {
                            "type" : "integer"
                        },
                        "anchor_conn_id" : {
                                "type" : "integer"
                        },
                        "mx_configuration_id" : {
                                "type" : "integer"
                        },
    },
                "client_param" : {
                        "connection_id" : {
                                "type" : "integer"
                        },
                        "adapt_param" : {
                            "type" : {"$ref" : "#definitions/adaptation_para
m" }
                        }
                }
        },
                        "adapt_param": {
                        "tunnel_ip_addr": {
                                "type": "string"
                        },
                        "tunnel_end_port": {
                                "type": "integer"
                        },
                        "shared_secret": {
                                "type": "string"
                        },
                        "mx_header_optimization": {
                                "type": "boolean",
                                "default": false
                        }
                },
                "delivery_connection": {
                        "connection_id": {
                                "$ref": "#definitions/connection_id"
```

```
                        },
                        "connection_type": {
                                "$ref": "#definitions/connection_type"
                        },
                        "adaptation_method": {
                                "$ref": "#definitions/adapt_method"
                        },
                        "adaptation_method_param": {
                                "$ref": "#definitions/adapt_param"
                        }
                },
                "app_madp_assoc": {
                        "anchor_conn_id" : {
                                "type" : "integer"
                        },
                        "mx_configuration_id" : {
                                "type" : "integer"
                        }

                        "ul_tft_list": {
                                "items": {
                                        "$ref": "#definitions/traffic_flow_t
emplate"
                                },
                                "type": "array"
                        },
                        "dl_tft_list": {
                                "items": {
                                        "$ref": "#definitions/traffic_flow_t
emplate"
                                },
                                "type": "array"
                        }
                }
                "predict_param_name": {
                        "enum": [
                                "validity time",
                                "bandwidth",
                                "jitter",
                                "latency",
                                "signal_quality"
                        ],
                        "type": "string"
                },
                "predict_add_param_name": {
                        "enum": [
                                "WLAN_RSSI",
                                "WLAN_LOAD",
                                "LTE_RSRP",
                                "LTE_RSRQ",
```

```
                                "NR_RSRP",
                                "NR_RSRQ"
                            ],
                            "type": "string"
                    }
            },
            "id": "http://www.ietf.org/mams/definitions.json"
}
```

C.3.3. MX Discover

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_discover.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"}
  },
  "type": "object"
}
```

C.3.4. MX System Update

```
{
        "$schema": "http://json-schema.org/draft-04/schema#",
        "additionalProperties": false,
        "id": "http://www.ietf.org/mams/mx_system_info.json",
        "properties": {
                "message_type": {
                        "$ref": "mx_base_def.json#/message_type_def"
                },
                "sequence_num": {
                        "$ref": "mx_base_def.json#/sequence_num_def"
                },
                "version": {
                        "$ref": "mx_base_def.json#/version_def"
                },
                "ncm_connections": {
                        "type": "array",
                        "items": [
                                { "$ref": "definitions.json#/connection" },
                                { "$ref": "definitions.json#/ncm_end_point"
}
                        ]
                }
        },
        "type": "object"
}
```

C.3.5. MX Capability Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_capability_req.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
        "adaptation_methods": {
      "items": { "$ref" : "definitions.json#/adaptation_method"},
      "type": "array"
    },
    "anchor_connections": {
      "items": { "$ref" : "definitions.json#/connection"},
      "type": "array"
    },
    "convergence_methods": {
      "items": { "$ref" : "definitions.json#/convergence_method"},
      "type": "array"
    },
    "delivery_connections": {
      "items": { "$ref" : "definitions.json#/connection"},
      "type": "array"
    },
    "feature_active": {
      "items": { "$ref" : "definitions.json#/feature_status"},
      "type": "array"
    },
    "num_anchor_connections": {
      "type": "integer"
    },
    "num_delivery_connections": {
      "type": "integer"
    }
  },
  "type": "object"
}
```

C.3.6. MX Capability Response

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_capability_resp.json",
  "properties": {
        "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
    "adaptation_methods": {
      "items": { "$ref" : "definitions.json#/adaptation_method" },
      "type": "array"
    },
    "anchor_connections": {
      "items": { "$ref" : "definitions.json#/connection"},
      "type": "array"
    },
    "convergence_methods": {
      "items": { "$ref" : "definitions.json#/convergence_method" },
      "type": "array"
    },
    "delivery_connections": {
      "items": { "$ref" : "definitions.json#/connection"},
      "type": "array"
    },
    "feature_active": {
      "items": { "$ref" : "definitions.json#/feature_status"},
      "type": "array"
    },
    "num_anchor_connections": {
      "type": "integer"
    },
    "num_delivery_connections": {
      "type": "integer"
    },
        "unique_session_id" : {
                        "$ref": "definitions.json#/unique_session_id"
        }
  },
  "type": "object"
}
```

C.3.7. MX Capability Ack

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "definitions": {},
    "id": "http://www.ietf.org/mams/mx_capability_ack.json",
    "properties": {
                "message_type" : {"$ref": "mx_base_def.json#/message_type_de
f"},
                "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_de
f"},
                "version" : {"$ref": "mx_base_def.json#/version_def"},
                "unique_session_id" : { "$ref": "definitions.json#/unique_se
ssion_id" },
        "capability_ack": {"$ref" : "definitions.json#/capability_acknowledg
ement"}
    },
    "type": "object"
}
```

C.3.8. MX Reconfiguration Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://www.ietf.org/mams/mx_reconf_req.json",
  "properties": {
        "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
        "unique_session_id" : {
                        "$ref": "definitions.json#/unique_session_id"
        },
        "connection_id" : {"$ref" : "definitions.json#/connection_id" },
    "ip_address": {"$ref" : "definitions.json#/ip_address" },
    "mtu_size": {
          "maximum" : 65535,
      "minimum": 1,
      "type": "integer"
    },
        "ssid" : {
         "type" : "string"
        },
    "reconf_action": {
      "enum": [
            "release",
      "setup",
              "update"
      ],
      "id": "/properties/reconf_action",
      "type": "string"
    },
        "connection_status" : {"$ref" : "definitions.json#/connection_status
"},
        "delivery_node_id" : {"$ref": "definitions.json#/delivery_node_id"}
  },
  "type": "object"
}
```

C.3.9. MX Reconfiguration Response

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://www.ietf.org/mams/mx_reconf_rsp.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"}
  },
  "type": "object"
}
```

C.3.10. MX UP Setup Configuration

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "definitions": {
   "convergence_configuration" : {
               "mx_configuration_id": { "type" : "integer"},
               "convergence_method": { "$ref" : "definitions.json#/conv_met
hod" },
          "convergence_method_params": {
            "properties": {
              "proxy_ip": { "$ref" : "definitions.json#/ip_address" },
              "proxy_port": {"$ref" : "definitions.json#/port" },
              "client_key": {"$ref" : "definitions.json#/client_key" }
            },
            "type": "object"
          },
                 "num_delivery_connections": {
            "type": "integer"
          },
          "delivery_connections": {
            "items":{ "$ref" : "definitions.json#/delivery_connection" },
                          "type": "array"
          }
        }
  },
  "id": "http://www.ietf.org/mams/mx_up_setup_conf_req.json",
  "properties": {
        "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
    "num_anchor_connections": {
      "type": "integer"
    },
    "anchor_connections": {
```

```
      "items": {
        "properties": {
          "connection_id": { "$ref" : "definitions.json#/connection_id" },
          "connection_type": { "$ref" : "definitions.json#/connection_type"
},
                  "num_active_mx_conf" : { "type" : "integer" },
                  "convergence_config" : {
                        "items":{ "$ref" : "definitions/convergence_configur
ation" },
                        "type" : "array"
                  }

         },
        "type": "object"
      },
      "type": "array"
    }
  },
  "type": "object"
}
```

C.3.11. MX UP Setup Confirmation

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://www.ietf.org/mams/mx_up_setup_cnf.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
        "unique_session_id" : { "$ref": "definitions.json#/unique_session_id
" },
        "probe_param" : { "$ref": "definitions.json#/probe_param" },
        "num_delivery_conn" : {
                "type" : "integer"
        },
        "client_params" : {
                "type" : "array",
                "items" : [
                        {"$ref": "definitions.json#/client_param"}
                ]
        }
  },
  "type": "object"
}
```

C.3.12. MX Traffic Steering Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {
        "conn_list" : {
                "items" : { "$ref" : "definitions.json#/connection_id" },
                "type": "array"
        },
        "ul_delivery" : {
                "ul_tft" : { "$ref" : "definitions.json#/traffic_flow_templa
te"},
                "connection_list" : { "$ref" : "#definitions/conn_list" }
        }
  },
  "id": "http://www.ietf.org/mams/mx_traffic_steering_req.json",
  "properties": {
        "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
"connection_id": {"$ref" : "definitions.json#/connection_id" },
"mx_configuration_id": { "type" : "integer"},
   "downlink_delivery": {
      "items": { "$ref" : "definitions.json#/connection_id" },
      "type": "array"
    },
    "feature_activation": {
      "items": {"$ref" : "definitions.json#/feature_status" },
      "type": "array"
    },
        "default_uplink_delivery": {
                "type": "integer"
        },
    "uplink_delivery": {
      "items": { "$ref" : "#definitions/ul_delivery" },
      "type": "array"
    }
  },
  "type": "object"
}
```

C.3.13. MX Traffic Steering Response

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://example.com/example.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
        "unique_session_id" : { "$ref": "definitions.json#/unique_session_id
" },
        "feature_activation": {
      "items": {"$ref" : "definitions.json#/feature_status" },
      "type": "array"
    }
  },
  "type": "object"
}
```

C.3.14. MX Application MADP Association Request

```
{
        "$schema": "http://json-schema.org/draft-04/schema#",
        "definitions": {},
        "id": "http://example.com/example.json",
        "properties": {
                "message_type": {
                        "$ref": "mx_base_def.json#/message_type_def"
                },
                "sequence_num": {
                        "$ref": "mx_base_def.json#/sequence_num_def"
                },
                "version": {
                        "$ref": "mx_base_def.json#/version_def"
                },
                "unique_session_id": {
                        "$ref": "definitions.json#/unique_session_id"
                },
                "app_madp_assoc_list": {
                        "items": {
                                "$ref": "definitions.json#/app_madp_assoc"
                        },
                        "type": "array"
                }
        },
        "type": "object"
}
```

C.3.15. MX Application MADP Association Response

```
   {
          "$schema": "http://json-schema.org/draft-04/schema#",
          "definitions": {},
          "id": "http://example.com/example.json",
          "properties": {
                  "message_type": {
                          "$ref": "mx_base_def.json#/message_type_def"
                  },
                  "sequence_num": {
                          "$ref": "mx_base_def.json#/sequence_num_def"
                  },
                  "version": {
                          "$ref": "mx_base_def.json#/version_def"
                  },
                  "unique_session_id": {
                          "$ref": "definitions.json#/unique_session_id"
                  },
                  "is_success": {
                          "type": "boolean"
                  }
          },
          "type": "object"
   }
```

C.3.16. MX Path Estimation Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://www.ietf.org/mams/mx_path_est_req.json",
  "properties": {
        "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
    "active_probe_ack_req": {
      "enum": [
        "no",
                "yes"
      ],
      "type": "string"
    },
    "active_probe_freq_ms": {
          "maximum" : 10000,
      "minimum": 100,
          "type": "integer"
```

```
    },
    "active_probe_size_bytes": {
          "maximum": 1500,
      "minimum": 100,
      "type": "integer"
    },
        "active_probe_duration_sec" : {
          "maximum" : 100,
          "minimum" : 10,
          "type" :  "integer"
        },
    "connection_id": { "$ref" : "definitions#/connection_id" },
    "init_probe_ack_req": {
      "enum": [
        "no",
        "yes"
      ],
      "type": "string"
    },
    "init_probe_size_bytes": {
      "maximum": 1500,
      "minimum": 100,
      "type": "integer"
    },
    "init_probe_test_duration_ms": {
      "maximum": 10000,
      "minimum": 100,
      "type": "integer"
    },
    "init_probe_test_rate_Mbps": {
      "maximum": 100,
      "minimum": 1,
      "type": "integer"
    }
  },
  "type": "object"
}
```

C.3.17. MX Path Estimation Report

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://www.ietf.org/mams/mx_path_est_results.json",
  "properties": {
        "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
        "unique_session_id" : { "$ref": "definitions.json#/unique_session_id
" },
    "active_probe_results": {
      "properties": {
        "avg_tput_last_probe_duration_Mbps": {
          "maximum":100,
          "minimum": 1,
          "type": "number"
        }
      },
      "type": "object"
    },
    "connection_id": { "$ref" : "definitions.json#/connection_id" },
    "init_probe_results": {
      "properties": {
                "lost_probes_percentage": {
                  "maximum": 100,
          "minimum": 1,
          "type": "integer"
        },
       "probe_rate_Mbps": {
          "maximum": 100,
          "minimum": 1,
          "type": "number"
        }
      },
      "type": "object"
    }
  },
  "type": "object"
}
```

C.3.18. MX SSID Indication

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "definitions": {},
    "id": "http://www.ietf.org/mams/mx_ssid_indication.json",
    "properties": {
                "message_type" : {"$ref": "mx_base_def.json#/message_type_de
f"},
                "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_de
f"},
                "version" : {"$ref": "mx_base_def.json#/version_def"},
        "ssid_list": {
            "items": {
                                "properties" : {
                                        "ssid_type": { "$ref" : "definitions
.json#/ssid_types" },
                                        "ssid_id" : {
                                                "type" : "integer"
                                        }
                                }
            },
            "type": "array"
        }
    },
    "type": "object"
}
```

C.3.19. MX Measurements Configuration

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "definitions" : {
        "meas_conf" : {
                "connection_id" : { "$ref" : "definitions.json#/connection_i
d" },
                "connection_type" : { "$ref" : "definitions.json#/connection
_type" },
                "meas_rep_conf" : {
                        "items" : { "$ref" : "definitions.json#/meas_report_
conf" },
                        "type" : "array"
                }
        }
  },
  "id": "http://www.ietf.org/mams/mx_measurement_conf.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
        "measurement_configuration" : {
                "items" : {"$ref" : "#definitions/meas_conf" },
                "type" : "array"
        }
  },
  "type": "object"
}
```

C.3.20. MX Measurements Report

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "definitions": {},
    "id": "http://www.ietf.org/mams/mx_measurement_report.json",
    "properties": {
                "message_type" : {"$ref": "mx_base_def.json#/message_type_de
f"},
                "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_de
f"},
                "version" : {"$ref": "mx_base_def.json#/version_def"},
                "unique_session_id" : { "$ref": "definitions.json#/unique_se
ssion_id" },
        "measurment_reports": {
            "items": {
                "properties": {
                    "connection_id": {
                        "$ref" : "definitions.json#/connection_id"
                    },
                                        "connection_type" : {
                                                "$ref" : "definitions.json#/
connection_type"
                                        },
                                        "delivery_node_id" : {
                                                "$ref" : "definitions.json#/
delivery_node_id"
                                        },
                    "measurements": {
                        "items": {
                            "properties": {
                                "measurement_type": {
                                  "$ref" : "definitions.json#/meas_report_pa
ram"
                                },
                                "measurement_value": {
                                    "type": "integer"
                                }
                            },
                            "type": "object"
                        },
                        "type": "array"
                    }
                },
                "type": "object"
            },
            "type": "array"
        }
    },
    "type": "object"
}
```

C.3.21. MX Keep Alive Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_keep_alive_req.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
        "keep_alive_reason" : {"$ref": "definitions.json#/keep_alive_reason"
},
        "unique_session_id" : {"$ref": "definitions.json#/unique_session_id"
},
        "connection_id" : {"$ref": "definitions.json#/connection_id"},
        "delivery_node_id" : {"$ref": "definitions.json#/connection_id"}
  },
  "type": "object"
}
```

C.3.22. MX Keep Alive Response

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_keep_alive_rsp.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
        "unique_session_id" : {"$ref": "definitions.json#/unique_session_id"
}
  },
  "type": "object"
}
```

C.3.23. MX Session Termination Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_keep_alive_req.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
        "unique_session_id" : { "$ref": "definitions.json#/unique_session_id
" },
        "reason" : {
                "enum" : [
                        "MX_NORMAL_RELEASE",
                        "MX_NO_RESPONSE",
                        "INTERNAL_ERROR"
                ],
           "type" : "string"
        }
  },
  "type": "object"
}
```

C.3.24. MX Session Termination Response

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_session_termination_resp.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
        "unique_session_id" : { "$ref": "definitions.json#/unique_session_id
" }
  },
  "type": "object"
}
```

C.3.25. MX Network Analytics Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_network_analytics_req.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
        "unique_session_id" : { "$ref": "definitions.json#/unique_session_id
" },
        "params" : {
                "items" : {"$ref": "definitions.json#/predict_param_name"},
                "type" : "array"
                }
  },
  "type": "object"
}
```

C.3.26. MX Network Analytics Response

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "definitions" : {
   "ParamPredictions" : {
    "param_name" : {"$ref" : "definitions.json#/predict_param_name"},
        "additional_param" : {"$ref" : "definitions.json#/predict_add_param_
name"},
        "prediction" : { "type" : "integer" },
        "likelihood" : { "type" : "integer" },
 "validity_time" : { "type" : "integer" }

   },
   "MXAnalyticsList" : {
                "connection_id" : { "$ref" : "definitions.json#/connection_i
d" },
                "connection_type" : { "$ref" : "definitions.json#/connection
_type" },
                "predictions " : {
                        "items" : { "$ref" : "#definitions/ParamPredictions"
 },
                        "type" : "array"
                }
        }
  },
  "id": "http://www.ietf.org/mams/mx_network_analytics_resp.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"},
        "param_list" : {
                                "items" : {"$ref": "#definitions/MXAnalytics
List"},
                                "type" : "array"}
  },
  "type": "object"
}
```

C.4. Examples in JSON

C.4.1. MX Discover

```
   {
     "version" : "1.0",
     "message_type" : "mx_discover",
     "sequence_num" : 1
   }
```

C.4.2. MX System Update

```
  {
    "version" : "1.0",
    "message_type" : "mx_system_info",
    "sequence_num" : 2,
    "ncm_connections" : [
                          {
                                  "connection_id" : 0,
                                  "connection_type" : "lte",
                                  "ncm_end_point" : {
                                          "ip_address" : "192.168.1.10",
                                          "port" : 1234
                                  }
                          },
                          {
                                  "connection_id" : 1,
                                  "connection_type" : "wifi",
                                  "ncm_end_point" : {
                                          "ip_address" : "192.168.1.10",
                                          "port" : 1234
                                  }
                          }
          ]
  }
```

C.4.3. MX Capability Request

```
   {
     "version" : "1.0",
     "message_type" : "mx_capability_req",
     "sequence_num" : 3,
     "feature_active" : [
                   {
                          "feature_name" : "lossless_switching",
                          "active" : true
                   },
                   {
                          "feature_name" : "fragmentation",
                          "active" : false
                   }
     ],
     "num_anchor_connections" : 2,
     "anchor_connections" : [
                   {
                          "connection_id" : 0,
                          "connection_type" : "lte"
                   },
                   {
                          "connection_id" : 1,
```

```
                        "connection_type" : "wifi"
                }
    ],
    "num_delivery_connections" : 2,
    "delivery_connections" : [
                {
                        "connection_id" : 0,
                        "connection_type" : "lte"
                },
                {
                        "connection_id" : 1,
                        "connection_type" : "wifi"
                }
    ],
    "convergence_methods" : [
                {
                        "method" : "GMA",
                        "supported" : true
                },
                {
                        "method" : "MPTCP_Proxy",
                        "supported" : false
                }
    ],
    "adaptation_methods" : [
                {
                        "method" : "UDP_without_DTLS",
                        "supported" : false
                },
                {
                        "method" : "UDP_with_TLS",
                        "supported" : false
                },
                {
                        "method" : "IPSec",
                        "supported" : true
                },
                {
                        "method" : "Client_NAT",
                        "supported" : false
                }
    ]
  }
```

C.4.4. MX Capability Response

```
   {
     "version" : "1.0",
     "message_type" : "mx_capability_resp",
     "sequence_num" : 3,
     "feature_active" : [
                    {
                            "feature_name" : "lossless_switching",
                            "active" : true
                    },
                    {
                            "feature_name" : "fragmentation",
                            "active" : false
                    }
     ],
     "num_anchor_connections" : 2,
     "anchor_connections" : [
                    {
                            "connection_id" : 0,
                            "connection_type" : "lte"
                    },
                    {
                            "connection_id" : 1,
                            "connection_type" : "wifi"
                    }
     ],
     "num_delivery_connections" : 2,
     "delivery_connections" : [
                    {
                            "connection_id" : 0,
                            "connection_type" : "lte"
                    },
                    {
                            "connection_id" : 1,
                            "connection_type" : "wifi"
                    }
     ],
     "convergence_methods" : [
                    {
                            "method" : "GMA",
                            "supported" : true
                    },
                    {
                            "method" : "MPTCP_Proxy",
                            "supported" : false
                    }
     ],
```

```
     "adaptation_methods" : [
                    {
                            "method" : "UDP_without_DTLS",
                            "supported" : false
                    },
                    {
                            "method" : "UDP_with_TLS",
                            "supported" : false
                    },
                    {
                            "method" : "IPSec",
                            "supported" : true
                    },
                    {
                            "method" : "Client_NAT",
                            "supported" : false
                    }
     ],
     "unique_session_id" : {
                    "ncm_id" : 110,
                    "session_id" : 1111
     }
   }
```

C.4.5. MX Capability Ack

```
   {
     "version" : "1.0",
     "message_type" : "mx_capability_ack",
     "sequence_num" : 3,
     "unique_session_id" : {
                    "ncm_id" : 110,
                    "session_id" : 1111
     },
     "capability_ack" : "MX ACCEPT"
   }
```

C.4.6. MX Reconfiguration Request

```
   {
     "version" : "1.0",
     "message_type" : "mx_reconf_req",
     "sequence_num" : 4,
     "unique_session_id" : {
                  "ncm_id" : 110,
                  "session_id" : 1111
     },
     "reconf_action" : "setup",
     "connection_id" : 0,
     "ip_address" : "192.168.110.1",
     "ssid" : "SSID_1",
     "mtu_size" : 1300,
     "connection_status" : "connected",
     "delivery_node_id" : "2A12C"
   }
```

C.4.7. MX Reconfiguration Response

```
   {
     "version" : "1.0",
     "message_type" : "mx_reconf_rsp",
     "sequence_num" : 4
   }
```

C.4.8. MX UP Setup Configuration Request

```
{
        "version": "1.0",
        "message_type": "mx_up_setup_conf_req",
        "sequence_num": 5,
        "num_anchor_connections": 2,
        "anchor_connections": [{                          "connection_id": 1,
                        "connection_type": "wifi",
                        "num_active_mx_conf" : 2,
                        "convergence_config" : [
                        {
                                "mx_configuration_id" : 1,
                                "convergence_method": "GMA",
                                "convergence_method_params": {},
                                "num_delivery_connections": 2,
                                "delivery_connections": [{
                                                "connection_id": 0,
                                                "connection_type": "lte",
                                                "adaptation_method": "UDP_wi
thout_DTLS",
                                                "adaptation_method_param": {
                                                        "tunnel_ip_addr": "6
.6.6.6",
                                                        "tunnel_end_port": 9
999,
```

```
                                                "mx_header_optimizat
ion": true
                                        }
                                },
                                {
                                        "connection_id": 1,
                                        "connection_type": "wifi"
                                }
                        ]
                },
                {
                    "mx_configuration_id" : 2,
                        "convergence_method": "GMA",
                        "convergence_method_params": {},
                        "num_delivery_connections": 1,
                        "delivery_connections": [{
                                        "connection_id": 0,
                                        "connection_type": "lte",
                                        "adaptation_method": "UDP_wi
thout_DTLS",
                                        "adaptation_method_param": {
                                                "tunnel_ip_addr": "6
.6.6.6",
                                                "tunnel_end_port": 8
877
                                        }
                                }
                        ]
                }
          ]
        },
        {
                "connection_id": 0,
                "connection_type": "lte",
                "udp_port": 8888,
                "num_delivery_connections": 2,
                "delivery_connections": [{
                                "connection_id": 0,
                                "connection_type": "lte"
                        },
                        {
                                "connection_id": 1,
                                "connection_type": "wifi",
                                "adaptation_method": "UDP_without_DT
LS",
                                "adaptation_method_param": {
                                        "tunnel_ip_addr": "192.168.3
.3",
                                        "tunnel_end_port": "6000"
                                }
                        }
                ]
        }
    ]
```

```
}
```

C.4.9. MX UP Setup Confirmation

```
   {
     "version" : "1.0",
     "message_type" : "mx_up_setup_cnf",
     "sequence_num" : 5,
     "unique_session_id" : {
                   "ncm_id" : 110,
                   "session_id" : 1111
     },
     "probe_param" : {
          "probe_port" : 48700,
              "anchor_conn_id" : 0,
              "mx_configuration_id" : 1
     },
     "num_delivery_conn" : 2,
     "client_params" : [
                   {
                           "connection_id" : 0,
                           "adapt_param" : {
                                   "udp_adapt_port" : 51000
                           }
                   },
                   {
                           "connection_id" : 1,
                           "adapt_param" : {
                                   "udp_adapt_port" : 52000
                           }
                   }
     ]
   }
```

C.4.10. MX Traffic Steering Request

```
   {
     "version" : "1.0",
     "message_type" : "mx_traffic_steering_req",
     "sequence_num" : 6,
     "connection_id" : 0,
     "mx_configuration_id" : 1,
     "downlink_delivery" : [
           {
                   "connection_id" : 0
           },
           {
                   "connection_id" : 1
```

```
        }
    ],
    "default_uplink_delivery" : 0,
    "uplink_delivery" : [
        {
            "ul_tft" : {
                "remote_addr_mask" : "10.10.0.0/24",
                "local_addr_mask" : "192.168.0.0/24",
                "protocol_type" :  6,
                "loca_port_range" : {
                  "start" : 100,
                  "end" : 1000
                },
                "remote_port_range" : {
                        "start" : 100,
                        "end" : 1000
                },
                "traffic_class" : 20,
                "flow_label" : 100
            },
            "conn_list" : [
                {
                        "connection_id" : 1
                }
            ]
        },
        {
            "ul_tft" : {
                "remote_addr_mask" : "10.10.0.0/24",
                "local_addr_mask" : "192.168.0.0/24",
                "protocol_type" :  6,
                "local_port_range" : {
                  "start" : 2000,
                  "end" : 2000
                },
                "remote_port_range" : {
                        "start" : 100,
                        "end" : 1000
                },
                "traffic_class" : 20,
                "flow_label" : 50
            },
            "conn_list" : [
                {
                        "connection_id" : 1
                }
            ]
        }
```

```
     ],
     "feature_activation" : [
                    {
                            "feature_name" : "dl_aggregation",
                            "active" : true
                    },
                    {
                            "feature_name" : "ul_aggregation",
                            "active" : false
                    }
            ]
   }
```

C.4.11. MX Traffic Steering Response

```
   {
            "version": "1.0",
            "message_type": "mx_traffic_steering_rsp",
            "sequence_num": 6,
            "unique_session_id": {
                    "ncm_id": 110,
                    "session_id": 1111
            },
            "feature_activation": [{
                            "feature_name": "lossless_switching",
                            "active": true
                    },
                    {
                            "feature_name": "fragmentation",
                            "active": false
                    }
            ]
   }
```

C.4.12. MX Application MADP Association Request

```
{
        "version": "1.0",
        "message_type": "mx_app_madp_assoc_req",
        "sequence_num": 6,
        "unique_session_id": {
                "ncm_id": 110,
                "session_id": 1111
        },
        "app_madp_assoc_list": [{
                        "connection_id" : 0,
"mx_configuration_id" : 1,
                        "ul_tft_list": [{
                                "protocol_type": 17,
                                "local_port_range": {
                                        "start": 8888,
                                        "end": 8888
                                }
                        }],
                        "dl_tft_list": [{
                                "protocol_type": 17,
                                "remote_port_range": {
                                        "start": 8888,
                                        "end": 8888
                                }
                        }]
                }
        ]
}
```

C.4.13. MX Application MADP Association Response

```
{
        "version": "1.0",
        "message_type": "mx_app_madp_assoc_resp",
        "sequence_num": 6,
        "is_success": true
}
```

C.4.14. MX Path Estimation Request

```
{
  "version" : "1.0",
  "message_type" : "mx_path_est_req",
  "sequence_num" : 7,
  "connection_id" : 0,
  "init_probe_test_duration_ms" : 100,
  "init_probe_test_rate_Mbps" : 10,
  "init_probe_size_bytes" : 1000,
  "init_probe_ack_req" : "yes",
  "active_probe_freq_ms" : 10000,
  "active_probe_size_bytes" : 1000,
  "active_probe_duration_sec" : 10,
  "active_probe_ack_req" : "no"
}
```

C.4.15. MX Path Estimation Results

```
{
  "version" : "1.0",
  "message_type" : "mx_path_est_results",
  "sequence_num" : 8,
  "unique_session_id" : {
                "ncm_id" : 110,
                "session_id" : 1111
  },
  "connection_id" : 0,
  "init_probe_results" : {
        "lost_probes_percentage" : 1,
        "probe_rate_Mbps" : 9.9
  },
  "active_probe_results" : {
        "avg_tput_last_probe_duration_Mbps" : 9.8
  }
}
```

C.4.16. MX SSID Indication

```
{
        "version" : "1.0",
        "message_type" : "mx_ssid_indication",
        "sequence_num" : 9,
        "ssid_list" : [
                {
                        "ssid_type" : "ssid",
                        "ssid_id" : "SSID_1"
                },
                {
                        "ssid_type" : "bssid",
                        "ssid_id" : "xxx-yyy"
                }
        ]
}
```

C.4.17. MX Measurements Configuration

```
{
        "version" : "1.0",
        "message_type" : "mx_measurement_conf",
        "sequence_num" : 10,
        "measurement_configuration" : [
        {
                "connection_id" : 0,
                "connection_type" : "wi-fi",
                "meas_rep_conf"  : [
                        {
                           "meas_rep_param" : "WLAN_RSSI",
                           "meas_threshold" : {
                                "high" : -10,
                                "low" : -15
                                },
                           "meas_period_ms" : 500
                        },
                        {
                           "meas_rep_param" : "WLAN_LOAD",
                           "meas_threshold" : {
                                "high" : -10,
                                "low" : -15
                                },
                           "meas_period_ms" : 500
                        },
                        {
                           "meas_rep_param" : "EST_UL_TPUT",
                           "meas_threshold" : {
                                "high" : 100,
                                "low" : 30
```

```
                                },
                            "meas_period_ms" : 500
                        }
                ]
        },
        {
                "connection_id" : 1,
                "connection_type" : "lte",
                "meas_rep_conf"  : [
                        {
                            "meas_rep_param" : "LTE_RSRP",
                            "meas_threshold" : {
                                 "high" : -10,
                                 "low" : -15
                                 },
                            "meas_period_ms" : 500
                        },
                        {
                            "meas_rep_param" : "LTE_RSRQ",
                            "meas_threshold" : {
                                 "high" : -10,
                                 "low" : -15
                                 },
                            "meas_period_ms" : 500
                        }
                ]
        }
    ]
   }
```

C.4.18. MX Measurements Report

```
{
        "version" : "1.0",
        "message_type" : "mx_measurement_report",
        "sequence_num" : 11,
        "unique_session_id" : {
                "ncm_id" : 110,
                "session_id" : 1111
    },
        "measurment_reports" : [
        {
                "connection_id" : 0,
                "connection_type" : "wi-fi",
                "delivery_node_id" : "2021A",
                "measurements"  : [
                        {
                           "measurement_type" : "WLAN_RSSI",
                           "measurement_value" : -12
                        },
                        {
                           "measurement_type" : "UL_TPUT",
                           "measurement_value" : 10
                        },
                        {
                           "measurement_type" : "EST_UL_TPUT",
                           "measurement_value" : 20
                        }
                ]
        },
        {
                "connection_id" : 1,
                "connection_type" : "lte",
                "delivery_node_id" : "12323",
                "measurements"  : [
                        {
                           "measurement_type" : "LTE_RSRP",
                           "measurement_value" : -12

                        },
                        {
                           "measurement_type" : "LTE_RSRQ",
                           "measurement_value" : -12

                        }
                ]
        }
  ]
}
```

C.4.19. MX Keep Alive Request

```
{
  "version" : "1.0",
  "message_type" : "mx_keep_alive_req",
  "sequence_num" : 12,
  "keep_alive_reason" : "Handover",
  "unique_session_id" : {
                "ncm_id" : 110,
                "session_id" : 1111
  },
  "connection_id" : 0,
  "delivery_node_id" : "2021A"
}
```

C.4.20. MX Keep Alive Response

```
{
  "version" : "1.0",
  "message_type" : "mx_keep_alive_rsp",
  "sequence_num" : 12,
  "unique_session_id" : {
                "ncm_id" : 110,
                "session_id" : 1111
  }
}
```

C.4.21. MX Session Termination Request

```
{
  "version" : "1.0",
  "message_type" : "mx_session_termination_req",
  "sequence_num" : 13,
  "unique_session_id" : {
                "ncm_id" : 110,
                "session_id" : 1111
  },
  "reason" : "MX_NORMAL_RELEASE"
}
```

C.4.22. MX Session Termination Response

```
{
  "version" : "1.0",
  "message_type" : "mx_session_termination_resp",
  "sequence_num" : 13,
  "unique_session_id" : {
                "ncm_id" : 110,
                "session_id" : 1111
  }
}
```

C.4.23. MX Network Analytics Request

```
{
     "version" : "1.0",
     "message_type" : "mx_network_analytics_req",
     "sequence_num" : 20,
     "unique_session_id" : {
                  "ncm_id" : 110,
                  "session_id" : 1111
     },
     "parmas" : [
         "jitter",
         "latency"
         ]
}
```

C.4.24. MX Network Analytics Response

```
{
        "version": "1.0",
        "message_type": "mx_network_analytics_resp",
        "sequence_num": 20,
        "param_list": [{
                        "connection_id": 1,
                        "connection_type": "wifi",
                        "predictions": [{
                                        "param_name": "jitter",
                                        "prediction": 100,
                                        "likelihood": 50,
                                        "validity_time": 10
                                },
                                {
                                        "param_name": "latency",
                                        "prediction": 19,
                                        "likelihood": 40,
                                        "validity_time": 10
                                }
                        ]
                },
                {
                        "connection_id": 2,
                        "connection_type": "lte",
                        "predictions": [{
                                        "param_name": "jitter",
                                        "prediction": 10,
                                        "likelihood": 80,
                                        "validity_time": 10
                                },
                                {
                                        "param_name": "latency",
                                        "prediction": 4,
                                        "likelihood": 60,
                                        "validity_time": 10
                                }
                        ]
                }
        ]
}
```

Appendix D. Definition of APIs provided by CCM to the Applications at the Client

This section provides an example implementation of the APIs exposed by the CCM to the Applications on the client, documented with OpenAPI using Swagger 2.0.

```
{
  "swagger": "2.0",
  "info": {
    "version": "1.0.0",
    "title": "Client Connection Manager (CCM)",
    "description": "API provided by CCM towards Application on a MAMS client
."
  },
  "host": "MAMS.ietf.org",
  "basePath": "/ccm/v1.0",
  "schemes": [
    "https"
  ],
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/capabilities": {
      "get": {
        "description": "This API can be used by application to request for c
apabilities of the CCM.",
        "produces": [
          "application/json",
          "text/html"
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/capability"
            }
          },
          "default": {
            "description": "unexpected error",
            "schema": {
              "$ref": "#/definitions/errorModel"
            }
          }
        }
      }
    },
    "/app_requirements": {
      "post": {
        "description": "This API is used by N-MADP to report any kind of MAM
S user specific errors to NCM.",
        "produces": [
          "application/json",
          "text/html"
```

```
        ],
        "parameters": [
          {
            "name": "app-requirements",
            "in": "body",
            "required": true,
            "schema": {
              "$ref": "#/definitions/app-requirements"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "OK"
          },
          "default": {
            "description": "unexpected error",
            "schema": {
              "$ref": "#/definitions/errorModel"
            }
          }
        }
      }
    },
    "/predictive_link_params": {
      "get": {
        "description": "This API is used by applications to get the informat
ion about predicted parameters for each delivery connection.",
        "produces": [
          "application/json",
          "text/html"
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/link-params"
            }
          },
          "default": {
            "description": "unexpected error",
            "schema": {
              "$ref": "#/definitions/errorModel"
            }
          }
        }
      }
    }
  },
```

```
  "definitions": {
    "connection-id": {
      "type": "integer",
      "format": "uint8"
    },
    "connection-type": {
      "enum": [
        "wi-fi",
        "5g-nr",
        "multi-fire",
        "lte"
      ],
      "type": "string"
    },
    "features": {
      "enum": [
        "lossless_switching",
        "fragmentation",
        "concatenation",
        "uplink_aggregation",
        "downlink_aggregation",
        "measurement"
      ],
      "type": "string"
    },
    "adaptation-methods": {
      "enum": [
        "UDP_without_DTLS",
        "UDP_with_DTLS",
        "IPSec",
        "Client_NAT"
      ],
      "type": "string"
    },
    "convergence-methods": {
      "enum": [
        "GMA",
        "MPTCP_Proxy",
        "GRE_Aggregation_Proxy",
        "MPQUIC"
      ],
      "type": "string"
    },
    "connection": {
      "type": "object",
      "properties": {
        "conn-id": {
          "$ref": "#/definitions/connection-id"
```

```
        },
        "conn-type": {
          "$ref": "#/definitions/connection-type"
        }
      }
    },
    "convergence-parameters": {
      "type": "object",
      "properties": {
        "conv-param-name": {
          "type": "string"
        },
        "conv-param-value": {
          "type": "string"
        }
      }
    },
    "convergence-details": {
      "type": "object",
      "properties": {
        "conv-method": {
          "$ref": "#/definitions/convergence-methods"
        },
        "conv-params": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/convergence-parameters"
          }
        }
      }
    },
    "capability": {
      "type": "object",
      "properties": {
        "connections": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/connection"
          }
        },
        "features": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/features"
          }
        },
        "adapt-methods": {
          "type": "array",
```

```
          "items": {
            "$ref": "#/definitions/adaptation-methods"
          }
        },
        "conv-methods": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/convergence-details"
          }
        }
      }
    },
    "qos-param-name": {
      "enum": [
        "jitter",
        "latency",
        "bandwidth"
      ],
      "type": "string"
    },
    "qos-param": {
      "type": "object",
      "properties": {
        "qos-param-name": {
          "$ref": "#/definitions/qos-param-name"
        },
        "qos-param-value": {
          "type": "integer"
        }
      }
    },
    "port-range": {
      "type": "object",
      "properties": {
        "start": {
          "type": "integer"
        },
        "end": {
          "type": "integer"
        }
      }
    },
    "protocol-type": {
      "type": "integer"
    },
    "stream-features": {
      "type": "object",
      "properties": {
```

```
        "proto": {
          "$ref": "#/definitions/protocol-type"
        },
        "port-range": {
          "$ref": "#/definitions/port-range"
        },
        "traffic-qos": {
          "$ref": "#/definitions/qos-param"
        }
      }
    },
    "app-requirements": {
      "type": "object",
      "properties": {
        "num-streams": {
          "type": "integer"
        },
        "stream-feature": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/stream-features"
          }
        }
      }
    },
    "param-name": {
      "enum": [
        "bandwidth",
        "jitter",
        "latency",
        "signal_quality"
      ],
      "type": "string"
    },
    "additional-param-name": {
      "enum": [
        "lte-rsrp",
        "lte-rspq",
        "nr-rsrp",
        "nr-rsrq",
        "wifi-rssi"
      ],
      "type": "string"
    },
    "link-parameter": {
      "type": "object",
      "properties": {
        "connection": {
```

```
          "$ref": "#/definitions/connection"
        },
        "param": {
          "$ref": "#/definitions/param-name"
        },
        "additional-param": {
          "$ref": "#/definitions/additional-param-name"
        },
        "prediction": {
          "type": "integer"
        },
        "likelihood": {
          "type": "integer"
        },
        "validity_time": {
          "type": "integer"
        }
      }
    },
    "link-params": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/link-parameter"
      }
    },
    "errorModel": {
      "type": "object",
      "description": "Error indication containing the error code and message
.",
      "required": [
        "code",
        "message"
      ],
      "properties": {
        "code": {
          "type": "integer",
          "format": "int32"
        },
        "message": {
          "type": "string"
        }
      }
    }
  }
}
```

Appendix E. Implementation Example using Python for MAMS Client and Server

E.1. Client Side Implementation

A simple client side implementation using python can be as following:

```
#!/usr/bin/env python
import asyncio
import websockets
import json
import ssl
import time
import sys

context = ssl.SSLContext(ssl.PROTOCOL_TLS)
context.verify_mode = ssl.CERT_REQUIRED
context.set_ciphers("RSA")
context.check_hostname = False
context.load_verify_locations("/home/mecadmin/certs/rootca.pem")

discoverMsg = {'version':'1.0',
'message_type':'mx_discover'}

MXCapabilityRes = { 'version':'1.0',
'message_type':'mx_capability_res',
'FeatureActive':[{'feature_name':'fragmentation', 'active':'yes'}, {'feature
_name':'lossless_switching', 'active':'yes'}],
'num_anchor_connections':1,
'anchor_connections':[{'connection_id':0, 'connection_type':'lte'}],
'num_delivery_connections':1,
'delivery_connections':[{'connection_id':1, 'connection_type':"wifi"}],
'convergence_methods':[{'method':'GMA', 'supported':'true'}],
'adaptation_methods':[{'method':'client_nat', 'supported':'false'}]
}

async def hello():
    async with websockets.connect('wss://localhost:8765', ssl=context) as we
bsocket:
     try:
      loopFlag=False
      while True:
          await websocket.send(json.dumps(discoverMsg))
          json_message = await websocket.recv()
          message = json.loads(json_message)
          if "message_type" in message.keys():
            print("Recieved message:{}".format(message["message_type"]),"ver
sion:{}".format(message["version"]))
            if message["message_type"] == "mx_capability_req" :
                await websocket.send(json.dumps(MXCapabilityRes))
                loopFlag=True
                while(loopFlag==True):
                      pass
     except:
          print("Client stopped")

asyncio.get_event_loop().run_until_complete(hello())
```

E.2. Server Side Implementation

A server client side implementation using python can be as following:

```
#!/usr/bin/env python
import asyncio
import websockets
import json
import ssl

ctx = ssl.SSLContext(ssl.PROTOCOL_TLS)
#ctx.set_ciphers("RSA-AES256-SHA")
ctx.load_verify_locations("/home/mecadmin/certs/rootca.pem")
certfile = "/home/mecadmin/certs/server.pem"
keyfile = "/home/mecadmin/certs/serverkey.pem"
ctx.load_cert_chain(certfile, keyfile, password=None)

MXCapabilityReq = { 'version':'1.0',
'message_type':'mx_capability_req',
'FeatureActive':[{'feature_name':'fragmentation', 'active':'yes'}, {'feature
_name':'lossless_switching', 'active':'yes'}],
'num_anchor_connections':1,
'anchor_connections':[{'connection_id':0, 'connection_type':'lte'}],
'num_delivery_connections':1,
'delivery_connections':[{'connection_id':1, 'connection_type':"wifi"}],
'convergence_methods':[{'method':'GMA', 'supported':'true'}],
'adaptation_methods':[{'method':'client_nat', 'supported':'false'}]
}


async def hello(websocket, path):
 try:
   while True:
     name = await websocket.recv()
     msg = json.loads(name)
     if "message_type" in msg.keys():
        print("Recieved message:{}".format(msg["message_type"]),"version:{}"
.format(msg["version"]))
        if msg['message_type'] == 'mx_discover':
           await websocket.send(json.dumps(MXCapabilityReq))

 except:
      print("client disconnected")

try:
 start_server = websockets.serve(hello, 'localhost', 8765,ssl=ctx)

 asyncio.get_event_loop().run_until_complete(start_server)
 asyncio.get_event_loop().run_forever()
except:
 print("server stopped")
```

Authors' Addresses

Satish Kanugovi
Nokia

Email: satish.k@nokia.com

Florin Baboescu
Broadcom

Email: florin.baboescu@broadcom.com

Jing Zhu
Intel

Email: jing.z.zhu@intel.com

Julius Mueller
AT&T

Email: jm169k@att.com

SungHoon Seo
Korea Telecom

Email: sh.seo@kt.com