

## 1. 序言

这个文档介绍 AP 控制管理协议(AP Control and Manage Protocol, ACAMP)，一个基于 SDN 架构和 Openflow 协议的标准的、互操作协议，它使 SDN 控制器(Controller)能够控制管理无线接入点(Access Point, AP)。

### 1.1 需求

部署和管理无线网络中 AP 的传统方法通常是采用 HTTP 或是通过特定第二层专有方法登录至 AP 并人工静态配置，在对于大型无线网络的部署和管理时，配置人员需要通过每个 AP 专有的配置方法逐个对 AP 进行配置，这种方法显得十分低效、不实用。

SDN 架构的出现，使得集中式的网络架构又被重新提起。SDN 的典型架构共分为三层，最上层为应用层，包括各种不同的业务和应用；中间的控制层主要负责处理数据平面资源的编排，维护网络拓扑，状态信息等；最下层的基础设施层负责数据处理、转发和状态收集。应用层与控制层之间的接口通常被称为北向接口，控制层与基础设施层之间的接口通常被称为南向接口。其中南向接口上现在应用最广泛的协议是 Openflow 协议，Openflow 协议制定了一个远程控制器控制和管理一个有线的 Openflow 交换机的规范。但是 Openflow 协议没有制定一个远程控制器控制和管理一个无线 AP 的规范，随着基于 SDN 架构的 WLAN 网络的出现，这种需求迫切的需要被解决。

### 1.2 目标

本文档扩展了 SDN 架构中南向接口上的 Openflow 协议，在其之上制定了 ACAMP 协议，ACAMP 协议的目标是：

1. 为 SDN 架构的 WLAN 网络中的 AP 提供控制与管理功能，功能包括对 AP 运行的无线参数进行配置，以及关联于 AP 的 STA 进行控制。
2. 对 SDN 架构的 WLAN 网络中的 AP 运行状况进行监测，统计数据收集。
3. 为了应对未来无线网络参数的增加和变更，采用了通用的封装和传输机制以获得良好的可扩展性。

ACAMP 协议仅关注 Controller 和 AP 之间的接口。Controller 之间、STA 到 Controller 之间的通信以及 AP 对配置的应用严格的讲超出了本文档的范围。

### 1.3 术语

SDN 控制器(Controller)

SDN 架构中定义的设备之一，主要功能是负责将应用层的具体应用需求解析，并且将这些网络需求发送给底层的数据转发层的设备上。在本文档中扮演在控制平面上对 AP 进行控制和管理的网络实体。

#### 无线接入点(Access Point, AP)

包括射频天线和无线物理层的设备，主要功能是通过预定的无线参数建立无线网络，并发送和接收该无线网络中 STA 的流量，在本文档中扮演受到控制的网络实体。

#### 站(Station, STA)

同样是包括射频天线和无线物理层的设备，与某 AP 建立关联加入特定的无线网络后，向该 AP 发送数据以及从该 AP 上接收数据，是用户侧的网络实体。

#### Openflow 交换机(Openflow Switch)

SDN 架构中定义的设备之一。与传统交换机不同，Openflow 交换机以一种流表的方式进行数据的转发处理，网络中的每一个流均要由 Openflow 交换机中的流表规则来控制处理，流表由控制器下发或本地静态配置。在本文档中扮演 Controller 和 AP 之间的中间设备。

## 2. 协议综述

ACAMP 协议是基于 Openflow 协议的高层协议，在 Controller 和 AP 的控制平面上进行通信，协议使用 UDP 的 6606 端口，协议对等实体是 Controller 与 AP，中间实体可能会存在一个或多个 Openflow 交换机。

ACAMP 协议从可选的 Discovery 阶段开始。AP 发送 Discovery Request 消息，诱发任何收到该消息的 Controller 用 Discovery Response 消息进行响应。AP 从收到的消息发现 Controllers，并向其中一个 Controller 发起注册控制管理服务请求。一旦 Controller 同意了 AP 的注册请求，双方进入运行阶段。

在运行阶段，ACAMP 协议提供从 Controller 到 AP 的一系列控制指令，这些指令有用于查询、更改 AP 配置，也有用于管理与 AP 进行通信的 STA，包括收集 AP 与这些 STA 间通信的统计信息。同时 ACAMP 协议也为 Controller 提供机制以获取由 AP 自己收集的统计信息。

ACAMP 协议提供心跳消息(Keep-Alive)机制，该机制用于维持 Controller 和 AP 间的 ACAMP 通信信道。如果该通道发生中断，AP 将断开与当前 Controller 的连接，并可能去尝试发现新的 Controller。

### 2.1 运行场景

ACAMP 协议定义了 Controller 与 AP 之间的交互，协议对等实体是 Controller 与 AP。但目前市面上并不存在支持 Openflow 协议的 AP，为了兼容市面上绝大部分设备以确保协议的实际可操作性，协议当中把 AP 等同为目前集中式 WLAN 组网下的 FIT AP。

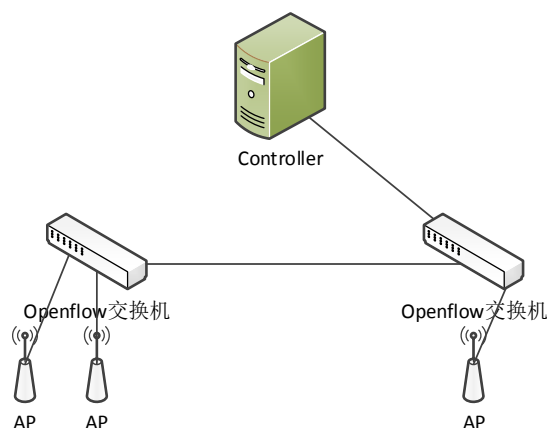


图 2-1 ACAMP 协议运行场景

图 2-1 为 ACAMP 协议运行场景图。Openflow 交换机上可以连接多个 AP，这些 AP 为不同地区的 STA 提供接入服务，而这些移动站点之间的数据通过 Openflow 交换机向核心网转发。Controller 可以集中控制管理每个连接在 Openflow 交换机上的 AP。

在该场景图中，Controller 和 Openflow 交换机之间会建立 Openflow 链路，在这段链路中 ACAMP 协议承载在 Openflow 协议之上；在 Openflow 交换机与 AP 之间，ACAMP 协议采用 UDP 进行承载。

## 2.2 协议承载

虽然 ACAMP 协议定义的协议对等实体是 Controller 和 AP，但 Controller 和 AP 不是直接相连，中间需要经过 Openflow 交换机。Openflow 协议作为 Controller 与 Openflow 交换机之间的接口协议，为 Controller 与 Openflow 交换机的交互提供了通路。因此，所有抵达 Controller 的消息必须打包成 Openflow 消息，这一过程由 Openflow 交换机完成。这导致了从 AP 发送给 Controller 的消息在 Controller 与 Openflow 交换机之间以及交换机与 AP 之间两段通路上承载的方式有所差异，在 Openflow 交换机与 AP 之间 ACAMP 消息直接通过 UDP 协议承载，而在 Controller 与交换机之间整个 UDP 报文会再由 Openflow 协议进一步封装。协议封装过程如图 2-2 所示。

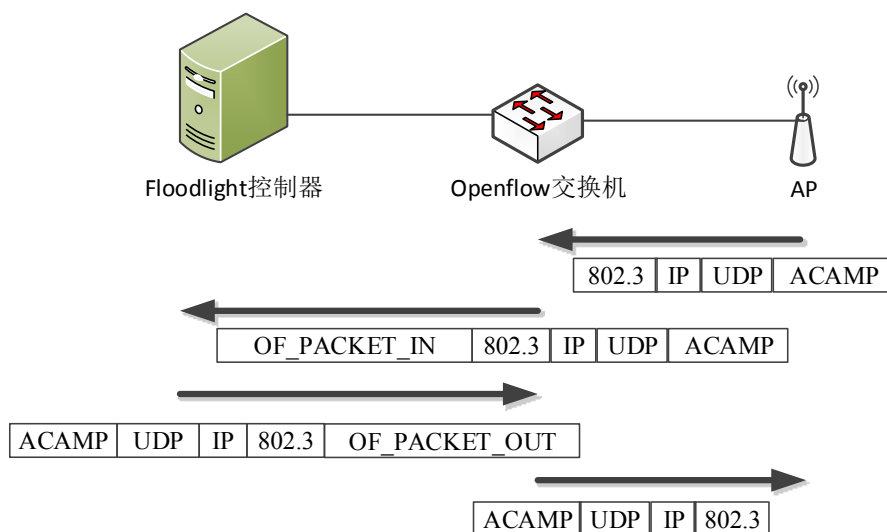


图 2-2 协议承载

在 Openflow 交换机到 Controller 的链路上 ACAMP 协议采用 Openflow 协议进行承载。具体的，发往 Controller 的方向上选用 Packet-In 消息作为承载，Controller 收到来自 Openflow 交换机打包上来的 Packet-In 消息（封装着 ACAMP 消息），经过处理，下行中使用 Packet-Out 消息承载并发往 Openflow 交换机，同时指定将 ACAMP 消息从指定的交换机端口发出。Openflow 交换机会将 Openflow 头部去掉，将里面封装的 ACAMP 消息利用 UDP 承载送往 AP。

2.3 会话建立综述

Controller 与 AP 的会话可以分为两个大阶段，注册服务前和注册服务后。前者是指 AP 还没有向 Controller 注册控制管理服务，Controller 与 AP 之间并没有相互关联；后者是指 AP 已经向 Controller 注册控制管理服务并得到响应，Controller 与 AP 之间已经相互关联。

整个会话建立过程如图 2-3 所示。

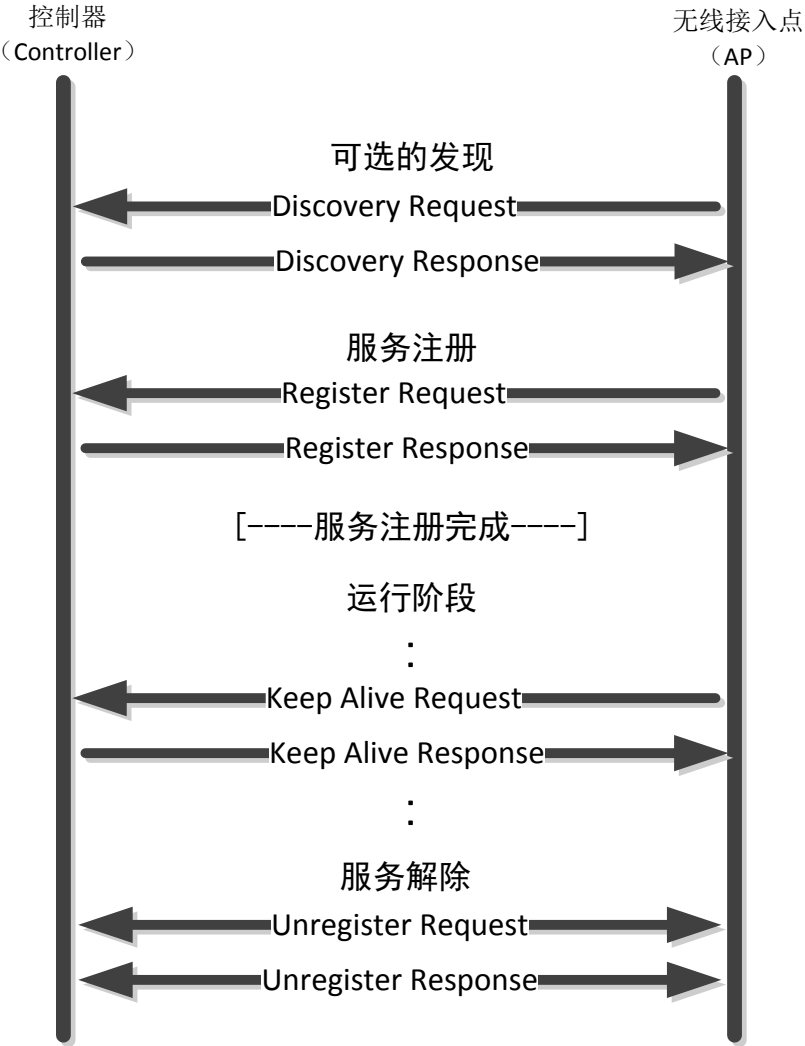


图 2-3 ACAMP 会话建立过程

ACAMP 会话从可选的发现过程开始，注册服务前，AP 可能不知道当前网络当中是否存在 Controller 以及 Controller 所处的网络位置，因此 AP 需要探测当前网络当中的所有潜在的 Controller。这个过程在 ACAMP 协议中称为 Controller 的发现过程。发现过程是一个可选的

过程，AP 也可以在本地配置文件中预设 Controller 的网络位置。

通过发现过程，AP 可以得到网络当中所有 Controller 的信息，随后按照一定的策略，选择某一个 Controller 并向其注册控制管理服务。当然，AP 也可以根据预设的 Controller 位置直接向 Controller 直接发起服务注册。这个过程在 ACAMP 协议中称为 AP 服务注册过程。

注册服务后，AP 与 Controller 建立了关联关系，双方进入运行状态。在运行阶段，Controller 可以通过协议定期收集 AP 的统计数据 and AP 的运行状态，为自动化决策提供一些依据。Controller 还可以动态的修改 AP 的配置信息，以及动态管理与 AP 相关联的 STA。

注册服务的解除是双方均可发起的，Controller 可依据自己的决策解除与 AP 的关联关系。另外考虑到在多控制器情况下，AP 可能需要作 Controller 的切换工作，因此 AP 也可能主动与已关联的 Controller 中脱离，这个过程在 ACAMP 协议中定义为 AP 服务解除过程。

每个会话过程中具体交互的消息将在 4 小节中进一步叙述。

## 2.4 状态机定义

下述状态图表示整个 AP、Controller 生命周期的状态及可能的状态变化。由于 ACAMP 协议本身不是一个十分复杂的协议，所以将 AP 和 Controller 的状态机集成在一起，状态机如图 2-4 所示，触发 ACAMP 状态机中状态转换的事件将在 2.4.2 小节中进行说明。

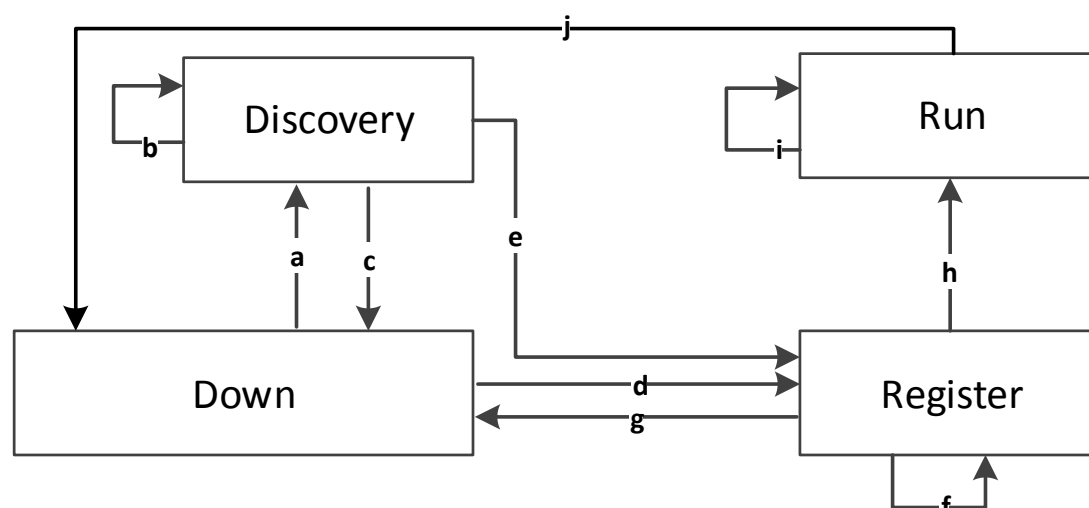


图 2-4 ACAMP 协议集成的状态机

如上所述，ACAMP 协议状态机由 Controller 和 AP 共用。如果这些状态存在不被共享情况，即对于 Controller 或 AP 中的一方是不合法的状态转移，将在下面介绍的转换时显式标出。对于每一种定义的状态，仅准许发送和接收某些消息，除了规定的消息，其余都是不合法的消息，每个阶段合法的消息将在 4 小节中详细进行介绍。

### 2.4.1 状态机实例

因为 AP 仅与单个 Controller 通信，所以 AP 仅需要单个 ACAMP 状态机实例。而在 Controller 上状态机的工作与此不同。因为 Controller 要与许多 APs 进行通信，因此 Controller 需要为每一个注册了服务的 AP 建立一个状态机实例。为了实现这一机制 Controller 可以使

用以下 3 个线程概念。值得注意的是，这里使用线程术语没有暗示实现必须使用的线程，只是提供实现 Controller 状态机的一种可选方法。

**Listener Thread:** Controller 的 Listener 线程通过监听 ACAMP 消息，主要用于处理 AP 的注册服务请求。该线程在 Controller 启动时创建，当一个新的注册服务请求到达且 Controller 接纳了这一请求后，Listener 线程将创建与 AP 关联的特定 Service 线程，同时向新线程传递 AP 的前后状态信息。

**Discovery Thread:** Controller 的 Discovery 线程专门用于负责接收和响应 Discovery Request 消息。Discovery 线程不记录任何每个 AP 的状态或其它信息。另外，Controller 必须采用一定策略保护自己，抵御一些未认证帧的攻击，或是拒绝一些含有潜在危险的 AP。

**Service Thread:** Controller 的 Service 线程用于处理每个 AP 的状态，Controller 为每个已注册服务的 AP 保持这样的线程。这个线程由 Listener 线程在某个到达的 AP 通过 Register 状态时创建。创建时，Service 线程继承来自 Listener 线程以及该 AP 的状态机副本。当与 AP 的通信完成或出现错误终止时，Service 线程终止并且释放所有相关联的资源。

#### 2.4.2 协议状态转换

本节介绍状态机中的状态转换，以及诱发它们的事件。本节所介绍的状态及状态转换是提供状态实现的一个参考，实际实现上可以与之略有差异。本节中有关重传机制的说明详见 2.5 小节，有关计时器的说明详见 3.5 小节，计数器最大值的相关说明详见 3.6 小节。

需要特别说明的是，在 Controller 的状态转换中，若没特别说明，均指每个 AP 独有的 Service Thread 中的状态转换。事实上，Discovery Thread 只会出现 Down 与 Discovery 状态间的相互转换，而 Listener Thread 只会出现 Down 与 Register 状态间的相互转换。

##### Down 到 Discovery(a):

**AP:** 为了防止大量 AP 同时启动引起发现阶段的广播风暴，AP 在静默 0 至 SilentInterval 间任意一个随机整数时间后传送第一个 Discovery Request，在传送第一个 Discovery Request 消息前，这个转换发生。一旦进入这个状态，AP 启动 DiscoveryInterval 计时器并设置 DiscoveryCount 计数器为 0。

**Controller:** 这个状态转换由 Controller 的 Discovery Thread 执行，当收到 Discovery Request 消息时发生。Controller 应当立即用 Discovery Response 消息进行响应，之后回到 Down 状态（详见 c 状态转换）。

##### Discovery 到 Discovery(b):

**AP:** 当 DiscoveryInterval 计时器到期，这个转换发生。AP 重传 Discovery Request 消息并重置 DiscoveryInterval 计时器，同时清除在前一个 Discovery 阶段收到的来自 Controllers 的所有消息。这个事件每转换一次，AP 都自增一次 DiscoveryCount 计数器。

**Controller:** 对于 Controller，这是个不合法的状态转换。

##### Discovery 到 Down(c):

**AP:** 当 DiscoveryInterval 计时器到期并且 DiscoveryCount 变量等于 MaxDiscovery 变量时，这个转换发生。一旦进入这个状态，AP 必须清除所有计时器和计数器，接着静默 0 至 SilentInterval 任意一个随机整数时间并忽略所有收到的 ACAMP 协议消息，在 SilentInterval 计时器结束之后，AP 可以重新开始 Discovery 阶段。

**Controller:** 这个状态转换由 Controller 的 Discovery Thread 执行，当发送了 Discovery

Response 消息后就立即执行该状态转换。

#### Down 到 Register(d):

**AP:** 该状态转换表示 AP 设置了静态的 Controller 地址并跳过了发现阶段。在传送第一个 Register Request 消息前，这个转换发生。AP 应为 Register Request 消息设定重传机制（详见 2.5 小节）。

**Controller:** 这个状态转换由 Controller 的 Listener Thread 执行，当收到 Register Request 消息时发生。Controller 应当立即用 Register Response 消息进行响应，之后回到 Down 状态（详见 g 状态转换）。若同意该 AP 注册，则还要为该 AP 建立 Service Thread，Service Thread 继承来自 Listener Thread 并进入 Run 状态（详见 h 状态转换）。同时在 Service Thread 中为 Register Response 消息设定重传机制。

#### Discovery 到 Register (e):

**AP:** 当 AP 收到了 Discovery Response 消息并决定向该 Controller 注册，在传送第一个 Register Request 消息前，这个转换发生。AP 应为 Register Request 消息设定重传机制。

**Controller:** 对于 Controller，这是个不合法的状态转换。

#### Register 到 Register (f):

**AP:** Register Request 消息的重传机制被触发。

**Controller:** 对于 Controller，这是个不合法的状态转换。

#### Register 到 Down(g):

**AP:** 当 AP 收到了拒绝注册的 Register Response 消息或是在重传机制结束后仍未收到合法的 Register Response 消息时，这个转换发生。一旦进入这个状态，AP 必须清除所有计时器和计数器，接着静默 0 至 SilentInterval 任意一个随机整数时间并忽略所有收到的 ACAMP 协议消息，在 SilentInterval 计时器结束之后，AP 可以重新开始 Discovery 或 Register 阶段。

**Controller:**

这个状态转换由 Controller 的 Listener Thread 执行，当发送了 Register Response 消息后就立即执行该状态转换。

#### Register 到 Run(h):

**AP:** 当 AP 收到了同意注册的 Register Response 消息后执行该状态转换。

**Controller:** 这个状态转换由 Controller 的 Service Thread 执行，当同意 AP 注册后从 Listener Thread 继承了该 AP 的信息及状态后，立刻发生该状态转换，并设定 WaitKeepAlive 计时器。

#### Run 到 Run(i):

**AP:** 这是 AP 的正常运行状态，AP 应为该状态下发送的所有报文设定重传机制。在该阶段，AP 应在每次 KeepAliveInterval 计时器到期时向 Controller 发送 Keep Alive Request 消息。值得注意的是，当 AP 收到任意来自 Controller 的合法 Request 或是 Response 都应重置 KeepAliveInterval 计时器。

**Controller:** 这是 Controller 的正常运行状态（Service Thread），同样 Controller 应为该状态下发送的所有报文设定重传机制。当 Controller 收到 AP 的 Keep Alive Request 消息时重置 WaitKeepAlive 计时器。值得注意的是，当 Controller 收到任意来自 AP 的合法 Request 或是 Response 都应重置 WaitKeepAlive 计时器。

Run 到 Down(j):

**AP:** 下列两种情况之一发生时, 发生该状态转换。AP 必须清除所有计时器和计数器, 启动 0 至 SilentInterval 间随机整数的计时器并忽略所有收到的 ACAMP 协议消息。在 SilentInterval 计时器结束之后, AP 可以重新开始 Discovery 或 Register 阶段。

1. 当重传机制失效, 重传次数超过最大重传次数时, 详见 2.5 小节说明。

2. 收到了 Unregister Request 消息, 回复 Unregister Response 同意取消关联, 资源释放过程详见 4.10 小节。

**Controller:** 下列三种情况之一发生时发生该状态转换。Controller 销毁 AP 对应的 Service Thread, 并清除所有计时器和计数器。

1. 当重传机制失效, 重传次数超过最大重传次数时。

2. 当 WaitKeepAlive 计时器到期时。

3. 收到了 Unregister Request 消息, 回复 Unregister Response 同意取消关联, 资源释放过程详见 4.10 小节。

## 2.5 序列号与重传

ACAMP 协议是一款可靠的传输协议。所有消息都以 Request 和 Response 消息成对出现 (详见 3.3 小节)。控制首部 Sequence Number 字段 (序列号) 用于配对 Request 消息和 Response 消息, 在一组消息中 Sequence Number 字段应被设为同一个值。

初始序列号应由 AP 和 Controller 各自在 0 至  $2^{32}-1$  的范围内随机产生。AP 通过 Register Request 消息头部第一次携带 AP 的序列号告知 Controller, Controller 通过 Register Response 消息中的 Controller Next Sequence Number 消息元素第一次携带 Controller 的序列号告知 AP, 值得一提的是, Register Response 头部携带的序列号应与 Register Request 头部携带的序列号一致 (也就是使用 AP 传输的序列号)。序列号在发送方每一个 Request 收到合法的 Response 后自增 1。

AP 与 Controller 都需要维护一个自己下一个发送的 Request 序列号值以及等待收到的对方发送的下一个 Request 序列号值。

基于此, 重传机制被描述如下。除了 Discovery 消息, 所有消息必须设定重传机制。

当发送方没有收到用于确认的 Response 消息, 由 RetransmitInterval 计时器和 MaxRetransmit 变量决定是否需要重传原始的 Request 消息。RetransmitInterval 计时器用于首次 Request 重传, 以后每一次重传相同 Request 消息该计时器重传间隔时间加倍, 但是不超过 KeepAliveInterval 计时器时间的一半, 直到收到 Response 或重传次数 RetransmitCount 达到 MaxRetransmit。

一旦重传次数 RetransmitCount 达到 MaxRetransmit, 发送方的状态转移至 Down 状态, 并进行后续的一些处理, 详见 2.4.2 小节关于 j 状态转换的描述。

接收方必须记录最后收到的 Request 消息的序列号, 并缓存相应的 Response 消息。如果收到具有相同序列号的重传, 必须立刻重传缓存的 Response 消息, 但不需要重新处理该 Request。如果收到较旧的 Request 消息, 即该消息的序列号较小, 忽略该消息。如果收到较新的 Request 消息, 即该消息序列号较大, 按正常情况处理该消息并更新序列号, 缓存新的 Response 消息。

另一点必须要说明的是, 在任何情况下 Controller 和 AP 都仅有一个未解决的 Request,



发送方不能改变可能重传的 Retransmitted Request 消息，换句话说，发送方在一个消息还未收到确认消息前，不能启动下一个消息的发送过程，直到收到对应的 Response 消息。如果发送方没有这样做，可能会导致双方的序列号不同步。

3. ACAMP 分组格式

这一节包含 ACAMP 协议的分组格式。ACAMP 协议分组由一个公共的 ACAMP 分组首部，再加上零个或多个 ACAMP 消息元素组成。当前版本只定义了 Control 类型的 ACAMP 消息，即 ACAMP 协议不带有任何数据载荷。ACAMP 协议的分组首部在下面定义。

3.1 ACAMP 首部

所有 ACAMP 协议消息都携带一个公共的 ACAMP 分组首部，首部包含了对协议本身，包括协议版本、协议类型的描述。同时还包含了对所承载消息的类型和长度的描述。紧跟在首部之后的是一段变长数据，该变长数据可以为空，也可以由多个消息元素(Message Element, ME)构成。

0	15	16	31
Version	Type	APID	
Sequence Number			
Message Type		Message Len	
Reserved			
Message Elements[0...N]...			

图 3-1 ACAMP 分组首部格式

图 3-1 为 ACAMP 分组首部格式示意图，在 Control 类型的 ACAMP 消息中不携带任何数据载荷，换句话说，Control 类型的 ACAMP 消息的整个分组格式就是如图 3-1 所示的格式。每个 ACAMP 协议当中定义的消息都可以用此格式来描述。ME 的个数是可变的，因此消息长度也是变长的。

**Version:** 1 字节的版本号字段，用于表示该消息使用 ACAMP 协议版本号，这个规范使用的版本编号为 3。

**Type:** 1 字节的类型字段，用于标识消息为 ACAMP 承载的是控制消息还是用户数据，当前仅定义了控制消息。此字段作为保留字段，用于 ACAMP 协议的扩展。

0 - 控制消息

**APID:** 2 字节的 AP 标识符字段，标识所控制 AP 的编号，此 ID 由 Controller 唯一分配，值范围为 1-65535。此字段值为 0 当且仅当该消息为发现消息、注册请求消息。

**Sequence Number:** 4 字节的序列号字段，用于匹配 Controller 和 AP 间的完成某一操作的一组消息（详见 4 小节），一组消息中该字段设置的值应是一样的，此值在初始化时由双方自行随机产生，随后单调递增，以确保每一组消息可以被唯一标识。范围为 0-4294967295。

**Message Type:** 2 字节的消息类型字段，定义了 ACAMP 协议所有的消息类型。合法的消息类型将在 3.3 小节中描述。

**Message Len:** 2 字节的消息长度字段，定义了 ACAMP 消息的总长度，包含变长的消息元素部分。

**Reserved:** 4 字节的保留字段，用于未来 ACAMP 协议的扩展，当前该字段必须被全部置为 0。

**Message Elements:** 变长的消息元素队列，包含零个或多个变长的消息元素，每个消息元素采用如图 3-2 所示的 TLV 格式进行定义。消息元素将在 3.4 小节中进一步讨论。

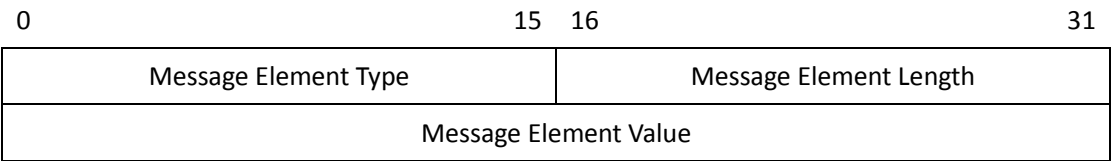


图 3-2 ACAMP 消息元素格式

3.2 控制消息

由于控制消息不携带有任何的数据载荷，因此控制消息的格式同 ACAMP 首部分组格式一样，如图 3-1 所定义。

3.3 消息类型

消息首部中的 Message Type 字段指出了 ACAMP 控制消息的功能，基于 ACAMP 的重传机制，消息类型总是以 2 个为一组成对的出现 (Request/Response)，在一组消息的交互中，所有报文必须携带相同的 Sequence Number。

每一组消息的类型值总是根据交互顺序递增的定义，因此当 Controller 或 AP 收到一个 Message Type 字段不能识别的消息时，将收到消息中的 Message Type 字段中的数加 1，并携带一个包含不识别该消息(Unrecognized Request)的 Result Code 消息元素，返回给所接收消息的发送方。Result Code 消息元素将在 3.4.1 小节中进一步进行说明。如果双方采用严格的协议实现及相同的协议版本号，可以忽略这一部分描述的实现。

消息类型在状态中的合法性以及其所需携带的消息元素参考 4 小节。

ACAMP Control Message Type 的合法取值在下表中规定：

Message Type	Message Type Value
Keep Alive Request	0x0001
Keep Alive Response	0x0002
Discovery Request	0x0003
Discovery Response	0x0004
Register Request	0x0101
Register Reponse	0x0102
Unregister Request	0x0103
Unregister Response	0x0104
Configuration Request	0x0201
Configuration Response	0x0202
Configuration Update Request	0x0203
Configuration Update Response	0x0204
//Scan Request	0x0301
//Scan Response	0x0302

<i>//Station Request</i>	<i>0x0303</i>
<i>//Station Response</i>	<i>0x0304</i>
<i>//Statistic Request</i>	<i>0x0305</i>
<i>//Statistic Response</i>	<i>0x0306</i>
System Request	0x0307
System Response	0x0308

### 3.4 消息元素类型

这一节定义包含在 ACAMP 协议控制消息中的消息元素(ME)。

消息元素用于携带控制消息中所需要的信息。每个消息元素由 **ME Type** 字段标识，如下面定义。消息元素的 **ME Length** 长度字段指出消息元素中 **ME Value** 的长度，即除去 **ME Type** 和 **ME Length** 字段剩余的长度。

本文档中所有消息元素使用类似于图 3-2 的格式来描述。为了简化本文档的介绍，接下来对消息元素介绍的格式图不包括首部字段(**ME Type** 和 **ME Length**)。

除非另有规定，控制消息不应指望这些消息元素以任何既定的次序出现。发送方可以以任何次序传送消息元素。除非另有说明，在给定的控制消息中每类消息元素只有一个。

当 **Controller** 或 **AP** 收到 **ACAMP** 消息，而该消息携带了按规定它不能携带的消息元素时，那么抛弃消息元素。

当 **Controller** 或 **AP** 收到不能识别的消息要素时，抛弃该消息元素。如果收到的消息是 **Request** 消息，在与该消息对应的 **Response** 消息携带相应的原因不能识别(**Unrecognized Message Element**)的 **Returned Message Element** 消息要素，包含不能识别的消息要素。如果双方采用严格的协议实现及相同的协议版本号，可以忽略这一部分描述的实现。

**ACAMP Control Message Element Type** 的合法取值在下表中规定：

Message Element Type	Message Element Type Value
Result Code	0x0001
Reason Code	0x0002
Assigned APID	0x0003
Discovery Type	0x0004
Registered Service	0x0005
Controller Name	0x0006
Controller Descriptor	0x0007
Controller IP Address	0x0008
Controller MAC Address	0x0009
AP Name	0x000a
AP Descriptor	0x000b
AP IP Address	0x000c
AP MAC Address	0x000d
Returned Message Element	0x000e
Controller Next Sequence Number	0x0010
Desired Configuration List	0x0011

SSID	0x0101
Channel	0x0102
Hardware Mode	0x0103
Suppress SSID	0x0104
Security Option	0x0105
MAC Filter Mode	0x0106
MAC Filter List	0x0107
Tx Power	0x0108
WPA Password	0x0202
Add MAC Filter List	0x0501
Delete MAC Filter List	0x0502
Clear MAC Filter List	0x0503
System Command	0x0401
<i>//Scanned WLAN Info</i>	<i>0x0402</i>
<i>//Station Info</i>	<i>0x0403</i>
<i>//Statistics</i>	<i>0x0404</i>

#### 3.4.1 Result Code

此消息元素是一个 2 字节的整型变量，指示了请求消息所对应的结果，通常包含在与请求消息所对应的回应消息中。支持下述枚举值：

0x0000 - 成功

0x0001 - 失败

0x0002 - 不识别的消息类型(Unrecognized Request)（如果双方采用严格的协议实现及相同的协议版本号，可以忽略这一部分描述的实现）

#### 3.4.2 Reason Code

此消息元素是一个 2 字节的整型变量，指示了失败的原因。支持下述枚举值：

0x0101 - 协议版本号不匹配（注册失败）

0x0102 - 资源枯竭（注册失败）

#### 3.4.3 Assigned APID

此消息元素是一个 2 字节的整型变量，指示了 Controller 为 AP 唯一指定的 APID 编号。注册服务后 AP 与 Controller 的通信，都要在消息首部携带这个唯一指定的 APID 编号。

#### 3.4.4 Discovery Type

此消息元素是一个 1 字节的整型变量，指示了 AP 发现 Controller 的方式。支持下述枚举值：

0 - 发现过程

1 - 静态配置

2 - 网关

#### 3.4.5 Registered Service

此消息元素是一个 1 字节的整型变量，指示了 AP 所要请求的服务类型。该消息元素的每 bit 对应一种提供的服务类型，将所要请求的服务类型对应的 bit 置 1，以 0bit 为低位，描述如下：

0bit – 控制管理服务（配置服务和站服务）

#### 3.4.6 Controller Name

此消息元素是一个 4-32 字节的变长字符串，指示了 Controller 的名称。

#### 3.4.7 Controller Descriptor

此消息元素是一个 1-128 字节的变长字符串，指示了对 Controller 的详细描述。

#### 3.4.8 Controller IP Address

此消息元素是一个 4 字节的二进制值，指示了 Controller 的 IPv4 地址。

#### 3.4.9 Controller MAC Address

此消息元素是一个 6 字节二进制值，指示了 Controller 的 MAC 地址

#### 3.4.10 AP Name

此消息元素是一个 4-32 字节变长的字符串，指示了 AP 的名称

#### 3.4.11 AP Descriptor

此消息元素是一个 1-128 字节变长的字符串，指示了对 AP 的详细描述。

#### 3.4.12 AP IP Address

此消息元素是一个 4 字节二进制值，指示了 AP 的 IPv4 地址

#### 3.4.13 AP MAC Address

此消息元素是一个 6 字节二进制值，指示了 AP 的 MAC 地址

#### 3.4.14 Returned Message Element

此消息元素是一个变长的字符串，指定需要返回的消息元素。每个子元素由一个 2 字节的 ME 类型编码和一个 2 字节的返回原因。返回原因有下列枚举值：

0x0000 -不识别的消息元素类型(Unrecognized Message Element)（如果双方采用严格的协议实现及相同的协议版本号，可以忽略这一部分描述的实现）

消息子元素格式如下：

0	15	16	31
Returned Message Element Type Value		Reason	
Returned Message Element Type Value		Reason	
...			

图 3-3 Returned Message Element 消息元素子格式

3.4.15 Controller Next Sequence Number

此消息元素是一个 4 字节的整型变量，指示了控制器下一个 Request 的序列号，在注册阶段告知 AP，以保证序列号同步。

3.4.16 Desired Configuration List

此消息元素是一个变长的字符串，指示需要查询的 AP 配置列表。该消息由 1 个或多个配置所对应的消息元素类型组成，每个类型长度 2 字节。合法的 Desired Configuration Msg Elem Type Value 包括 SSID(0x0101)、Channel(0x0102)、Hardware Mode(0x0103)、Suppress SSID(0x0104)、Security Option(0x0105)、Mac Filter Mode(0x0106)、MAC Filter List(0x0107)、Tx Power(0x0108)、WPA Password(0x0202)。格式如下：

0	15	31
Desired Configuration Msg Elem Type Value 1		Desired Configuration Msg Elem Type Value 2
Desired Configuration Msg Elem Type Value 3		Desired Configuration Msg Elem Type Value ...

图 3-4 Desired configuration List 消息元素子格式

3.4.17 SSID

此消息元素是一个 1-32 字节变长的字符串，指示了该无线网络的 SSID。

3.4.18 Channel

此消息元素是一个 1 字节的整型，指示了该无线网络所使用的无线信道，值范围为 1-13。

3.4.19 Hardware Mode

此消息元素是一个 1 字节的整型，指示该无线网络操作模式。可选操作模式有下列枚举值：

- 0 - a(IEEE 802.11a)
- 1 - b(IEEE 802.11b)
- 2 - g(IEEE 802.11g)
- 3 - n(IEEE 802.11n)

3.4.20 Suppress SSID

此消息元素是一个 1 字节的整型，指示对该 SSID 的隐藏操作，用于指定当接收到广播探测请求时是否发送空的 SSID，即是否广播 SSID 的操作。SSID 广播控制编码描述如下：

- 0 - 正常广播 SSID，回应所有广播的探测请求
- 1 - 发送空 SSID（长度为 0）的信标帧并忽略所有广播的探测请求

3.4.21 Security Option

此消息元素是一个 1 字节的整型，指示该无线网络安全类型。支持下列枚举值：

- 0 - None
- 2 - WPA/WPA2 Mixed
- 3 - WPA
- 4 - WPA2

3.4.22 MAC Filter Mode

此消息元素是一个 1 字节整型，指定 AP 的 MAC 访问控制列表类型。支持的枚举值如下：

- 0 - 不启用 MAC 访问控制
- 1 - 仅接受列表中列出的 MAC 地址的 STA 加入
- 2 - 拒绝列表中列出的 MAC 地址的 STA 加入

3.4.23 MAC Filter List

消息元素是一个变长的字符串，指示 MAC 访问控制列表中存在的地址。该消息可以为空，或由 1 个或多个 MAC 地址组成，每个 MAC 地址长度 6 字节。格式如下：

0	8	15	24	31
MAC Address 1				MAC Address 2
MAC Address 2		MAC Address ...		

图 3-5 MAC Filter List 消息元素子格式

3.4.24 Tx Power

此消息元素是一个 1 字节的整型，指示 AP 的传输功率，单位 dBm。

3.4.25 WPA Password

此消息元素是一个 8-63 字节变长字符串，指示 WPA 认证口令。

3.4.26 Add MAC Filter List

此消息元素是一个变长的字符串，指示向 MAC 访问控制列表添加 MAC 地址。该消息由 1 个或多个 MAC 地址组成，每个 MAC 地址长度 6 字节。格式如下：

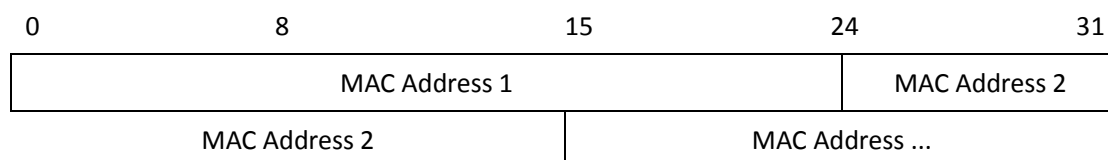


图 3-6 Add MAC Filter List 消息元素子格式

### 3.4.27 Delete MAC Filter List

此消息元素是一个变长的字符串，指示在 MAC 访问控制列表中删除 MAC 地址（不存在的将被忽略）。该消息由 1 个或多个 MAC 地址组成，每个 MAC 地址长度 6 字节。格式如下：

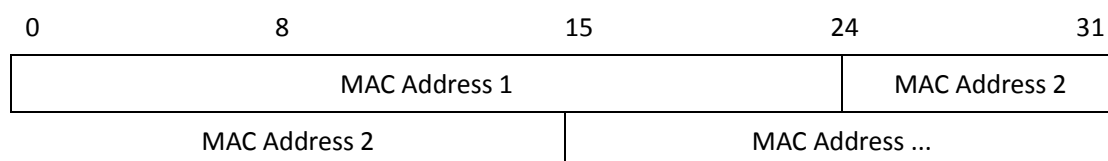


图 3-7 Delete MAC Filter List 消息元素子格式

### 3.4.28 Clear MAC Filter List

此消息元素不含任何内容长度为 0，指示清空 MAC 访问控制列表中所有 MAC 地址。

### 3.4.29 System Command

此消息元素是一个 1 字节整型，指定系统命令。支持的枚举值如下：

- 0 - 关闭 WLAN 功能
- 1 - 打开 WLAN 功能
- 2 - 重启 WLAN 功能
- 3 - 重启 Network
- //4 - 重启系统

### 3.4.30 Scanned WLAN Info

此消息元素指示扫描到的 WLAN 信息，是可以重复的消息元素。在目前协议版本中暂时还没有被定义。

### 3.4.31 Station Info

此消息元素指示关联的 Station 信息，是可以重复的消息元素。在目前协议版本中暂时还没有被定义。

### 3.4.32 Statistics

此消息元素指示 AP 的信息统计。在目前协议版本中暂时还没有被定义。

## 3.5 ACAMP 协议计时器



### 3.5.1 SilentInterval

AP 从 Down 状态启动发送 Discovery Request 或 Register Request 消息前，AP 必须静默最大的时间间隔。默认：20 秒。

### 3.5.2 DiscoveryInterval

AP 重传发送 Discovery Request 消息时的间隔时间。默认：10 秒。

### 3.5.3 RetransmitInterval

经过此时间间隔后发送方将重传没有得到确认的 ACAMP 消息，随着重传次数的增加，每一次重传该时间间隔会翻一倍，直到达到 KeepAliveInterval 的一半。默认：3 秒。

### 3.5.4 KeepAliveInterval

AP 发送给 Controller 的 Keep Alive Request 消息间的时间间隔，该计时器会被收到的来自对方的任何合法的 Request 或 Response 刷新（详见 2.4.2 小节关于 i 状态转换的描述）。默认：30 秒。

### 3.5.5 WaitKeepAlive

Controller 等待 AP 的 Keep Alive Request 消息的最长等待时间，该计时器会被收到的来自对方的任何合法的 Request 或 Response 刷新（详见 2.4.2 小节关于 i 状态转换的描述）。默认：60 秒。

## 3.6 ACAMP 协议变量

### 3.6.1 DiscoveryCount

AP 向 Controllers 广播 Discovery Request 消息的次数。初始值为 0，这是单调增加的计数器。

### 3.6.2 MaxDiscovery

AP 向 Controllers 广播 Discovery Request 消息的最大次数。默认：3 次。

### 3.6.3 RetransmitCount

发送方采用重传机制发送消息的次数。初始值为 0，这是单调增加的计数器。

### 3.6.4 MaxRetransmit

发送方采用重传机制发送消息的最大次数，超过后重传机制失效，发送方进入 Down 状态（详见 2.4.2 小节关于 j 状态转换的描述）。默认：5 次。

## 4. ACAMP 操作规范

这一节包含使用 ACAMP 协议完成某一操作时的交互规范。操作中的消息类型总是以 2 个为一组成对的出现 (Request/Response)，在一组消息的交互中，所有报文必须携带相同的 Sequence Number。除了发现操作，所有报文应设定重传机制。

除了下述规范中规定的消息元素，所有 Response 消息都可以携带 Returned Message Element 消息元素以标示不能识别的消息元素，也可能携带值为 0x0002 - 不识别的消息类型 (Unrecognized Request) 的 Result Code 消息元素以标示不能识别的消息。该描述详见 3.3 和 3.4 小节，如果双方采用严格的协议实现及相同的协议版本号，可以忽略这一部分描述的实现。

### 4.1 发现操作

AP 在与 Controller 关联之前，必须知道 Controller 所处的网络位置。在 IP 网络中，网络位置一般指 Controller 的 IP 地址。AP 有两种方式获取 Controller 的 IP 地址，一种是通过预配置的方式得到 Controller 的 IP 地址，包括静态配置、网关等方式，另外一种是采用广播的方式询问 Controller 的 IP 地址，ACAMP 协议把此过程称为 Controller 的发现操作。发现操作是可选的操作。

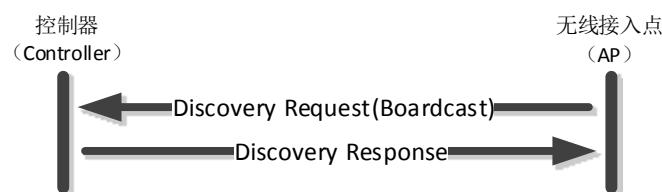


图 4-1 发现操作

#### 4.1.1 Discovery Request

Discovery Request 消息由 AP 发送，用于在网络中自动发现潜在的可用 Controllers，此消息的目的 IP 为 255.255.255.255，APID 字段置 0。

Discovery Request 消息由 AP 发送。Controller 不发送这条消息。

在 Discovery Request 消息中必须包括下述消息元素：无

在 Discovery Request 消息中可选包括下述消息元素：无

#### 4.1.2 Discovery Response

Discovery Response 消息用于 Controller 向 AP 通告自己所处的网络位置。一旦收到 Discovery Request 消息，Controller 立刻用 Discovery Response 消息进行响应，Discovery Response 消息的目的 IP 采用 Discovery Request 消息的源 IP 地址。Discovery Response 消息携带所有 Controller 相关的信息供 AP 决策。

Discovery Response 消息由 Controller 发送。AP 不发送这条消息。

在 Discovery Response 消息中必须包括下述消息元素：Controller Name、Controller Descriptor、Controller IP Address、Controller MAC Address

在 Discovery Request 消息中可选包括下述消息元素：无

## 4.2 服务注册操作

在获取 Controller 的 IP 后（预配置或者通过发现操作），AP 向 Controller 发送注册请求消息以注册所需的服务，Controller 需要响应 AP 的注册请求，一旦注册请求成功，AP 和 Controller 之间的控制管理关系才确立，否则，Controller 不会对此 AP 提供控制管理服务。服务注册后，AP 与 Controller 的通信必须在首部携带 Controller 分配的 APID。

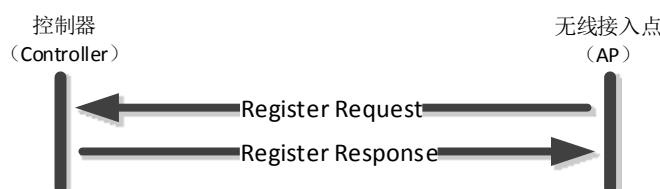


图 4-2 服务注册操作

### 4.2.1 Register Request

Register Request 由 AP 用于向 Controller 请求服务。在 AP 收到 Discovery Response 消息或已知 Controller 网络位置的情况下启动时，AP 发送 Register Request 消息，携带 AP 所要请求的控制管理服务及 AP 的所有信息。

Register Request 消息由 AP 发送。Controller 不发送这条消息。

在 Register Request 消息中必须包括下述消息元素：Registered Service、AP Name、AP Descriptor、AP IP Address、AP MAC Address

在 Register Request 消息中可选包括下述消息元素：Discovery Type

### 4.2.2 Register Response

Controller 需要对 AP 的注册请求进行响应，通过或拒绝。一旦注册请求通过，AP 和 Controller 之间的控制管理关系才确立，Controller 必须用此消息携带分配的 APID 及自己的下一个序列号，否则 Controller 不会对此 AP 提供控制管理服务。

Register Response 消息由 Controller 发送。AP 不发送这条消息。

若 Controller 同意 AP 注册：

在 Register Response 消息中必须包括下述消息元素：Result Code、Assigned APID、Registered Service、Controller Next Sequence Number、Controller Name、Controller Descriptor、Controller IP Address、Controller MAC Address

在 Register Response 消息中可选包括下述消息元素：无

若 Controller 拒绝 AP 注册：

在 Register Response 消息中必须包括下述消息元素：Result Code、Reason Code

在 Register Response 消息中可选包括下述消息元素：无

## 4.3 配置查询操作

关系建立后，Controller 可以对 AP 的配置进行查询，AP 对 Controller 所请求的配置如实进行回复。

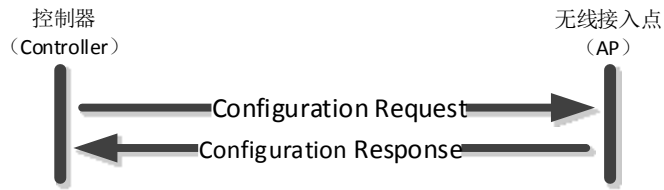


图 4-3 配置查询操作

#### 4.3.1 Configuration Request

Controller 通过 Configuration Request 消息查询 AP 的配置，该消息包含一个 Desired Configuration List 消息元素，该消息元素中仅包含需要查询的配置类型代码，无需查询的配置不必包含。

Configuration Request 消息由 Controller 发送。AP 不发送这条消息。

在 Configuration Request 消息中必须包括下述消息元素：Desired Configuration List

在 Configuration Request 消息中可选包括下述消息元素：无

#### 4.3.2 Configuration Response

AP 通过 Configuration Response 消息来回复它现有的工作参数，在该消息中 AP 应该将 Controller 询问的所有工作参数全部报告，供 Controller 记录和决策。

Configuration Report 消息由 AP 发送。Controller 不发送这条消息。

在 Configuration Report 消息中必须包括下述消息元素：无

在 Configuration Report 消息中可选包括下述消息元素：SSID、Channel、Hardware Mode、Suppress SSID、Security Option、Mac Filter Mode、MAC Filter List、Tx Power、WPA Password

### 4.4 配置更改操作

关系建立后，Controller 可以对 AP 的配置进行更改，AP 会自动使用更改后的配置重启网络，但不会返回配置是否成功应用的结果，Controller 应自行对配置进行合法性检查。

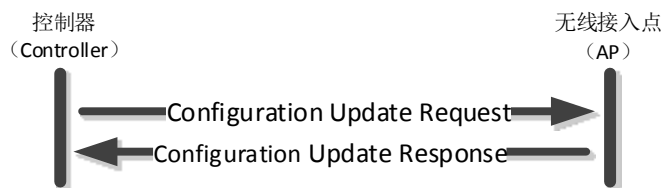


图 4-4 配置更改操作

#### 4.4.1 Configuration Update Request

Controller 通过 Configuration Request 消息更改配置，该消息包含一个或多个需要更改的配置对应的消息元素，无需更改的配置不必包含。

Configuration Update Request 消息由 Controller 发送。AP 不发送这条消息。

在 Configuration Update Request 消息中必须包括下述消息元素：无

在 Configuration Update Request 消息中可选包括下述消息元素：SSID、Channel、Hardware Mode、Suppress SSID、Security Option、Mac Filter Mode、MAC Filter List、Tx Power、WPA Password、Add MAC Filter List、Delete MAC Filter List、Clear MAC Filter List

#### 4.4.2 Configuration Update Response

AP 应用所有 Configuration Request 消息中包含的配置并重启网络。值得再次强调的是，AP 不会对配置进行任何检查，Controller 应自行对配置合法性进行检查，该回复仅表示 AP 收到了更改配置请求并尝试应用，不保证配置能够成功应用。

Configuration Update Response 消息由 AP 发送。Controller 不发送这条消息。

在 Configuration Update Response 消息中必须包括下述消息元素：无

在 Configuration Update Response 消息中可选包括下述消息元素：无

#### 4.5 扫描操作

关系建立后，Controller 可以命令 AP 对周围存在的 WLAN 网络进行扫描，AP 对扫描后回复所有扫描得到的 WLAN 信息。在目前的协议版本中，该操作暂时没有被定义。

##### 4.5.1 Scan Request

在目前的协议版本中，该消息暂时没有被定义。

##### 4.5.2 Scan Response

在目前的协议版本中，该消息暂时没有被定义。

#### 4.6 站信息操作

关系建立后，Controller 可以命令 AP 返回所关联的所有 STA 信息。在目前的协议版本中，该操作暂时没有被定义。

##### 4.6.1 Station Request

在目前的协议版本中，该消息暂时没有被定义。

##### 4.6.2 Station Response

在目前的协议版本中，该消息暂时没有被定义。

#### 4.7 统计信息操作

关系建立后，Controller 可以命令 AP 返回 AP 自身的统计信息。在目前的协议版本中，该操作暂时没有被定义。

##### 4.7.1 Statistic Request

在目前的协议版本中，该消息暂时没有被定义。

##### 4.7.2 Statistic Request

在目前的协议版本中，该消息暂时没有被定义。

#### 4.8 系统操作

关系建立后，Controller 可以命令 AP 对 WLAN、网络或是自身系统进行启动、断开、重启操作。值得注意的是，配置更改操作本身会引起 AP 重启 WLAN 一次，无需 Controller 再进行命令。另外，重启系统一般会导致控制管理服务断开，而且不会触发任何 Unregister 消息。

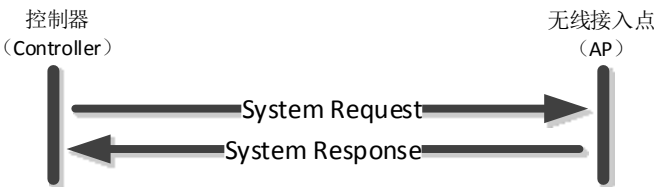


图 4-5 系统操作

##### 4.8.1 System Request

Controller 通过 System Request 消息命令 AP 进行系统操作，该消息包含一个 System Command 消息元素。

System Request 消息由 Controller 发送。AP 不发送这条消息。

在 System Request 消息中必须包括下述消息元素：System Command

在 System Request 消息中可选包括下述消息元素：无

##### 4.8.2 System Response

AP 执行 Controller 下发的系统操作命令。同样的，该 System Response 回复不会携带任何成功与否的字段，仅表示 AP 收到了这条系统操作命令的请求并尝试应用，不保证能否成功执行。

System Response 消息由 AP 发送。Controller 不发送这条消息。

在 System Response 消息中必须包括下述消息元素：无

在 System Response 消息中可选包括下述消息元素：无

#### 4.9 心跳保活操作

关系建立后，AP 每隔 KeepAliveInterval 需要向 Controller 发送 Keep Alive Request 消息以保证连接可靠，Controller 收到后立即使用 Keep Alive Response 消息回应。

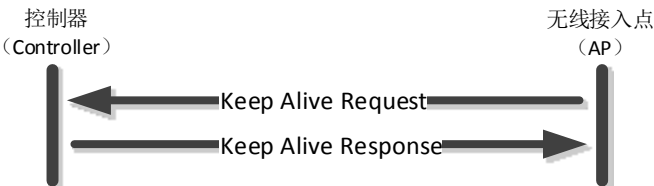


图 4-6 心跳保活操作

##### 4.9.1 Keep Alive Request

AP 每隔 KeepAliveInterval 时间向 Controller 进行连接确认，消息不携带任何的消息元素。  
Keep Alive Request 消息由 AP 发送。Controller 不发送这条消息。  
在 Keep Alive Request 消息中必须包括下述消息元素：无  
在 Keep Alive Request 消息中可选包括下述消息元素：无

#### 4.9.2 Keep Alive Response

Controller 收到 Keep Alive Request 后立即使用 Keep Alive Response 消息回应，以完成连接可靠的确认，消息不携带任何的消息元素。

Keep Alive Response 消息由 Controller 发送。AP 不发送这条消息。  
在 Keep Alive Response 消息中必须包括下述消息元素：无  
在 Keep Alive Response 消息中可选包括下述消息元素：无

#### 4.10 解除注册操作

关系建立后，双方均可主动通过 Unregister Request 消息向对方申请服务解除。发送方发送 Unregister Request 后并为其设定重传机制，同时将接收窗口设定为仅接收 Unregister Response 消息，在接收到 Unregister Response 或重传机制失效后释放资源；接收方收到 Unregister Request 后回应 Unregister Response 消息并释放相关资源。下图以 Controller 主动解除注册为例。

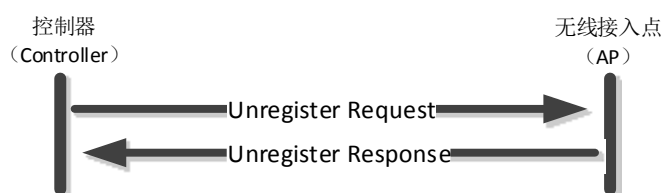


图 4-7 解除注册操作

##### 4.10.1 Unregister Request

解除注册请求由主动发起解除服务方发送，消息不携带任何的消息元素。  
Unregister Request 消息可由 Controller 或 AP 发起。  
在 Unregister Request 消息中必须包括下述消息元素：无  
在 Unregister Request 消息中可选包括下述消息元素：无

##### 4.10.2 Unregister Response

在收到 Unregister Request 后以 Unregister Response 回应，消息不携带任何的消息元素。  
Unregister Response 消息可由 Controller 或 AP 发起。  
在 Unregister Response 消息中必须包括下述消息元素：无  
在 Unregister Response 消息中可选包括下述消息元素：无