通信协议：Websocket 连接

基本信息

- 协议版本: 1
- 传输方式: Websocket
- 音频格式: OPUS
- 音频参数:
  - 采样率: 16000Hz
  - 通道数: 1
  - 帧长: 60ms

连接建立

1. 客户端连接Websocket服务器时需要携带以下headers:

```
Authorization: Bearer <access_token>
Protocol-Version: 1
Device-Id: <设备MAC地址>
Client-Id: <设备UUID>
```

设备MAC地址和UUID都是设备唯一识别码。

2. 连接成功后，客户端发送hello消息:

```
{
    "type": "hello",
    "version": 1,
    "transport": "websocket",
    "features": {
        "mcp": true
    },
```

```
    "audio_params": {
        "format": "opus",
        "sample_rate": 16000,
        "channels": 1,
        "frame_duration": 60
    }
}
```

3. 服务端响应hello消息:

```
{
    "type": "hello",
    "transport": "websocket",
    "audio_params": {
        "format": "opus",
        "sample_rate": 24000,
        "channels": 1,
        "frame_duration": 60
    }
}
```

Websocket协议不返回 session_id，所以消息中的会话ID可设置为空。

消息类型

1. 语音识别相关消息

开始监听

```
{
    "session_id": "<会话ID>",
    "type": "listen",
    "state": "start",
    "mode": "<监听模式>"
}
```

监听模式:

- "auto": 自动停止
- "manual": 手动停止
- "realtime": 持续监听

auto 与 realtime 是服务器端 VAD 的两种工作模式，realtime 需要 AEC 支持。

停止监听

```
{
    "session_id": "<会话ID>",
    "type": "listen",
    "state": "stop"
}
```

唤醒词检测

```
{
    "session_id": "<会话ID>",
    "type": "listen",
    "state": "detect",
    "text": "<唤醒词>"
}
```

2. 语音合成相关消息

服务端发送的TTS状态消息:

```
{
    "type": "tts",
    "state": "<状态>",
    "text": "<文本内容>" // 仅在 sentence_start 时携带
}
```

状态类型:

- "start": 开始播放
- "stop": 停止播放
- "sentence_start": 新句子开始

3. 中止消息

```
{
    "session_id": "<会话ID>",
    "type": "abort",
    "reason": "wake_word_detected" // 可选
}
```

4. MCP 相关消息

客户端 / 服务端:

```
{
    "session_id": "<会话ID>",
    "type": "mcp",
    "payload": <MCP Paylaod>
}
```

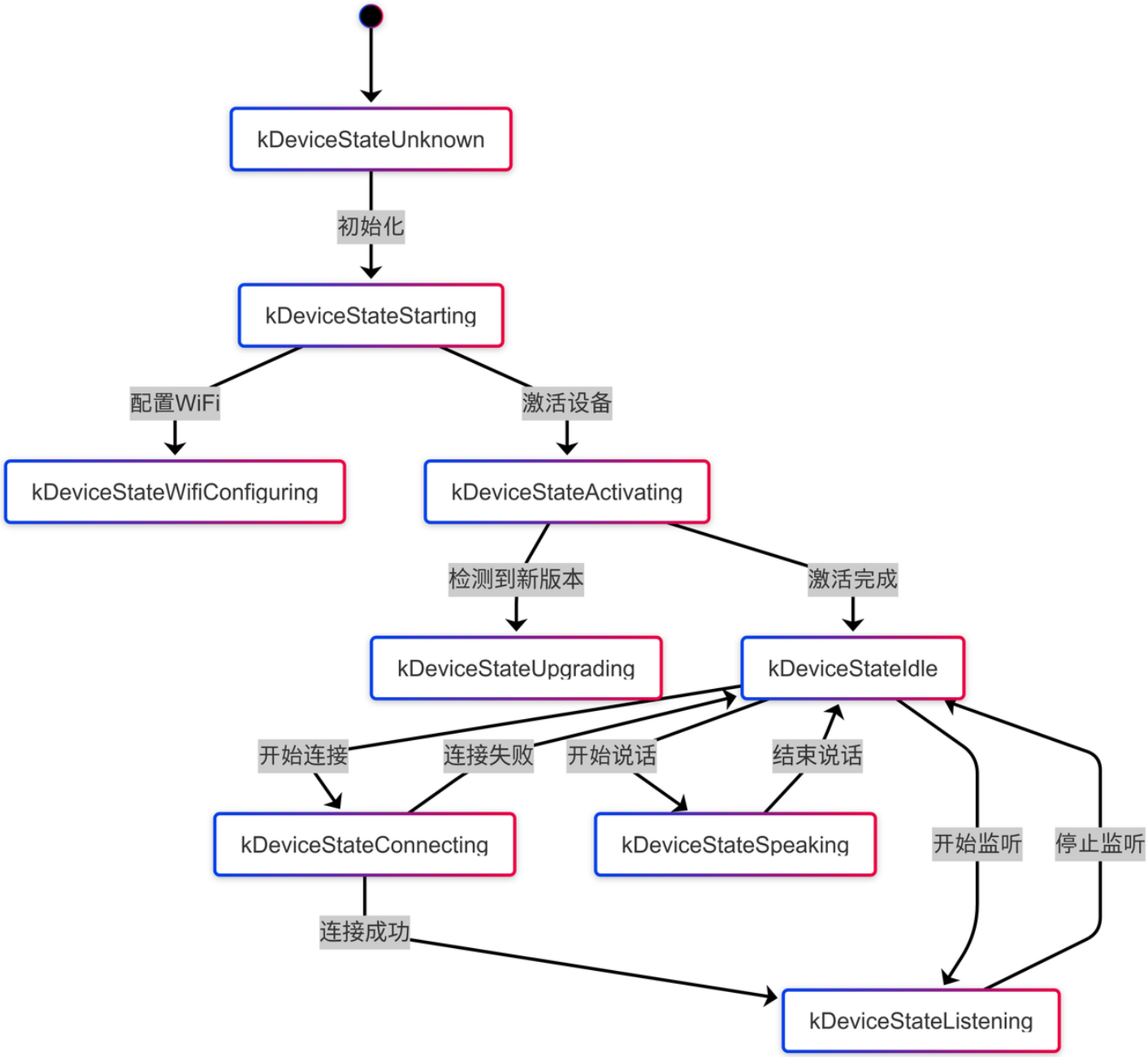
5. 情感状态消息

服务端发送:

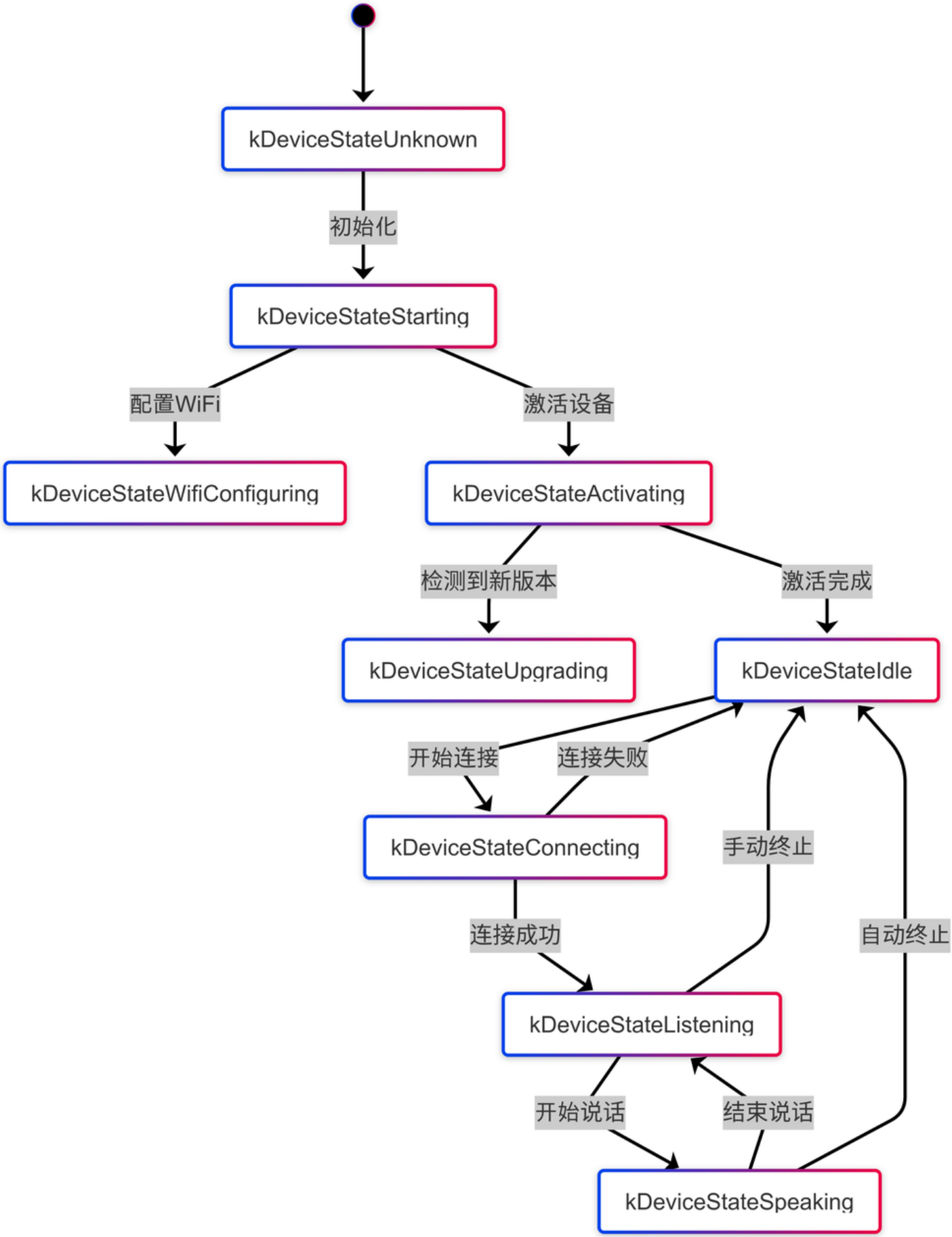
```
{
    "type": "llm",
    "emotion": "<情感类型>"
}
```

状态流程图

1. Manual 模式



2. Auto 模式


kDeviceStateUnknown
初始化
kDeviceStateStarting
配置WiFi
激活设备
kDeviceStateWifiConfiguring
kDeviceStateActivating
检测到新版本
激活完成
kDeviceStateUpgrading
kDeviceStateIdle
开始连接
连接失败
kDeviceStateConnecting
手动终止
连接成功
自动终止
kDeviceStateListening
开始说话
结束说话
kDeviceStateSpeaking

二进制数据传输

- 音频数据使用二进制帧传输
- 客户端发送OPUS编码的音频数据
- 服务端返回OPUS编码的TTS音频数据

错误处理

当发生网络错误时，客户端会收到错误消息并关闭连接。客户端需要实现重连机制。

会话流程

1. 建立Websocket连接
2. 交换hello消息
3. 开始语音交互:
    - 发送开始监听
    - 发送音频数据
    - 接收识别结果
    - 接收TTS音频
4. 结束会话时关闭连接