µD3TN 代码结构与数据处理流程

项目源码导读 · 目录职责、核心组件与 Bundle 收发主链

> µD3TN 是以 C 实现的延迟容忍网络（DTN）Bundle Protocol 守护进程，支持 BPv6/BPv7、多种收敛层（CLA）、AAP/AAP 2.0 应用接口和 SQLite 持久化。程序采用单进程、多任务架构，各组件通过队列交换事件，由 Bundle Processor（BPA）集中处理 bundle 状态和转发决策。

一、代码目录结构

```
ud3tn-master/
├── components/              C 核心实现
│   ├── daemon/              程序入口
│   ├── ud3tn/               BPA、bundle 管理、FIB、EID 等核心逻辑
│   ├── bundle6/ bundle7/    BPv6 / BPv7 解析、序列化、分片
│   ├── cla/                 网络收发适配层及 SQLite 存储 CLA
│   ├── aap/ aap2/           应用接口协议及 agent
│   ├── agents/              应用、echo、storage 等 agent
│   ├── routing/             兼容路由器及路由配置
│   ├── spp/                 CCSDS 空间包协议支持
│   └── platform/            POSIX 平台抽象：任务、队列、IO、时间等
├── include/                 C 接口和结构定义
├── generated/               Protobuf/Nanopb 等生成代码
├── external/                Nanopb、TinyCBOR、Unity 等外部依赖
├── pyd3tn/                  Python 版 BP/CLA 协议实现
├── python-ud3tn-utils/      AAP/AAP2 Python 客户端和命令行工具
├── tools/                   CLA 测试、分析等辅助工具
├── test/                    单元、集成和功能测试
├── doc/                     架构、协议、构建和存储文档
└── Makefile                 构建、测试、文档及代码检查入口
```

二、核心组件与职责

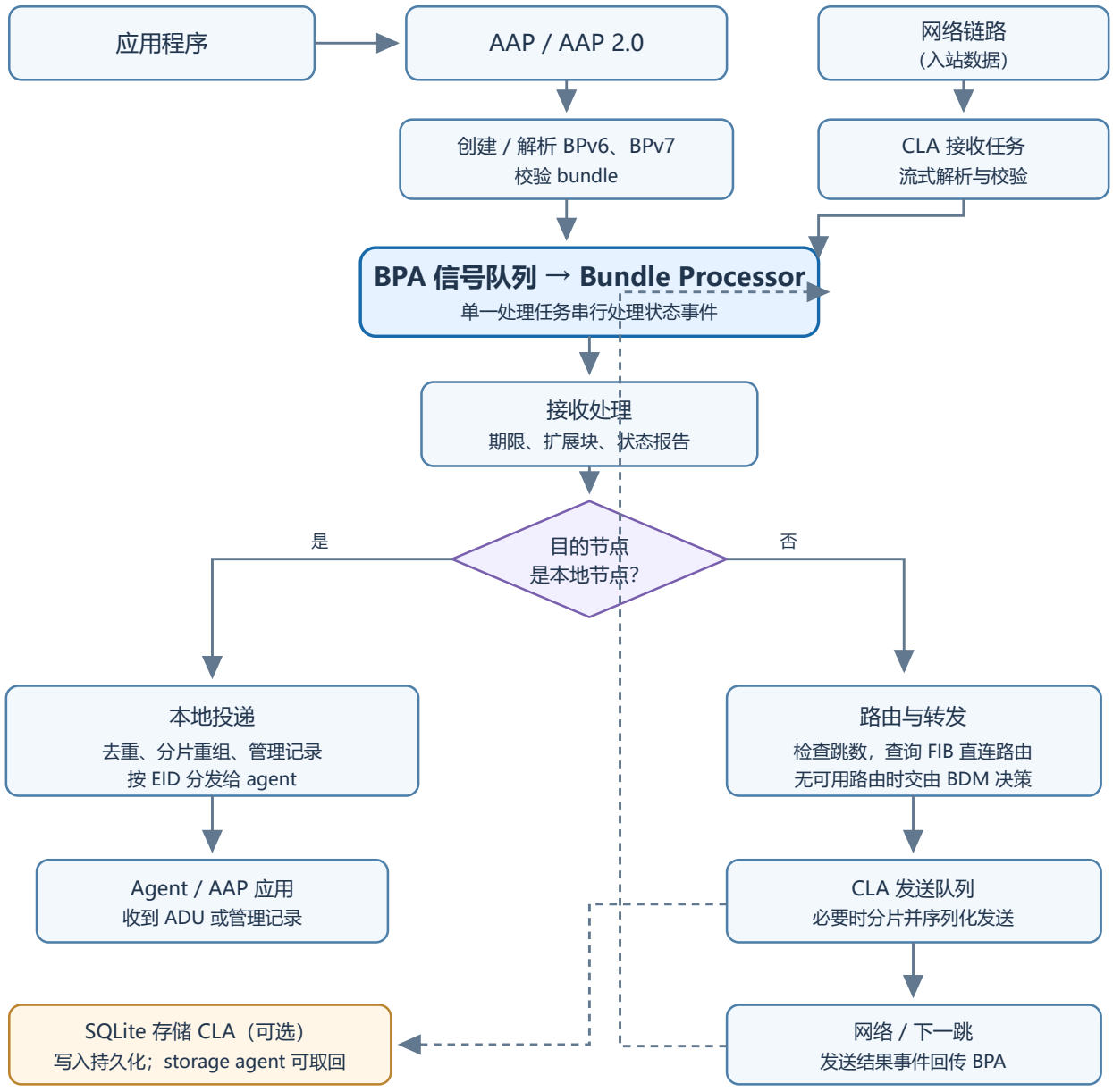
组件	职责	代码入口 / 重点位置
Daemon	解析启动选项，初始化平台及子系统，启动调度器。	`components/daemon/main.c` `components/ud3tn/init.c`
Bundle Processor（BPA）	接收所有 bundle 和链路事件；执行有效期、扩展块、去重、分片重组、本地投递与转发处理。	`components/ud3tn/bundle_processor.c`
BPv6 / BPv7	协议数据模型、创建、解析、序列化、报告与分片处理。	`components/bundle6/` `components/bundle7/`
CLA	抽象底层链路；负责连接、收发队列、流式解析/序列化。包含 MTCP、TCPCLv3、TCP-SPP、BIBE 和 SQLite。	`components/cla/`

AAP / AAP 2.0 与 Agents	向应用开放 bundle 收发；AAP 2.0 还支持链路/FIB 管理和转发决策。	`components/aap/`
		`components/aap2/`
		`components/agents/`

FIB / Routing / BDM	FIB 记录当前节点到 CLA 链路的映射；BDM 可通过 AAP 2.0 动态提供下一跳及分片建议。	`components/ud3tn/fib.c` `components/routing/`
Platform HAL	封装任务、队列、信号量、时间和 IO，支撑组件间并发通信。	`components/platform/`
Python 工具生态	pyD3TN 提供 Python 协议实现；ud3tn-utils 提供 AAP 客户端、BDM 和命令行工具。	`pyd3tn/` `python-ud3tn-utils/`

核心入口之间的关系：`main.c` → `init.c` 初始化任务和接口 → BPA 消费信号队列 → CLA / AAP / Agent 通过该队列向 BPA 传递事件。

三、Bundle 数据处理流程图



图中实线表示主要数据流；虚线表示发送结果事件返回 BPA，以及转发到 SQLite 存储 CLA 的可选路径。链路状态和 FIB 更新也通过 BPA 信号队列处理。

四、主链各阶段说明

1. **应用发包**：AAP/AAP2 接收 ADU，创建 BPv6/BPv7 bundle，发送本地分发信号给 BPA。参考 `components/aap2/aap2_agent.c`。
2. **链路收包**：CLA 的接收任务流式解析协议数据，验证 bundle，并用 `BP_SIGNAL_BUNDLE_INCOMING` 通知 BPA。参考 `components/cla/cla_contact_rx_task.c`。

3. **BPA 接收处理**：记录接收时间，检查生命周期和扩展块；随后根据目的 EID 进入本地投递或转发。参考 `components/ud3tn/bundle_processor.c` 的 `bundle_receive()`、`bundle_dispatch()`。
4. **本地投递**：检测重复 bundle；必要时执行分片重组和管理记录解析，再根据目标 EID 找到 agent 并交付 ADU。没有接收 agent 时按错误路径丢弃或报告。
5. **路由与分片**：转发前检查跳数，优先使用 FIB 中的直连路由；无可用直连路由时交由 BDM/AAP2 提供下一跳或分片方案。参考 `send_bundle()`。
6. **CLA 发送**：发送任务更新转发字段，将 bundle 流式序列化并写入网络或存储后端；完成时向 BPA 报告发送成功或失败。
7. **状态收敛**：BPA 接收发送结果事件，更新引用计数和 bundle 状态，决定成功、重新分发或报告转发失败。

五、SQLite 持久化路径

SQLite 不是 BPA 内嵌的一份特殊存储代码，而是作为 CLA 提供。BDM 或 FIB 可将 bundle 指向 SQLite CLA；SQLite CLA 将序列化数据写入数据库。storage agent 可按目的 EID 查询、删除或 push 取回 bundle。取回后的 bundle 重新进入 BPA 处理队列，整体上与经网络 CLA 收到的 bundle 类似。持久化的详细使用方式见项目的 `doc/sqlite-storage.md`。

六、建议阅读顺序

1. `components/daemon/main.c`：进程入口。
2. `components/ud3tn/init.c`：任务、队列、CLA、AAP/AAP2 和路由器的初始化顺序。
3. `include/ud3tn/bundle_processor.h`：BPA 信号、组件间传递的数据结构。
4. `components/ud3tn/bundle_processor.c`：事件分发、bundle 接收、本地投递及转发。
5. `components/cla/cla_contact_rx_task.c` 与 `cla_contact_tx_task.c`：链路收发和编解码流程。
6. `components/aap2/aap2_agent.c`、`components/ud3tn/fib.c`、`components/cla/storage/cla_sqlite.c`：应用入口、路由状态和持久化实现。