知更：TSN 工业网络平台

清华大学
http://tns.thss.tsinghua.edu.cn/ziggo/

目录

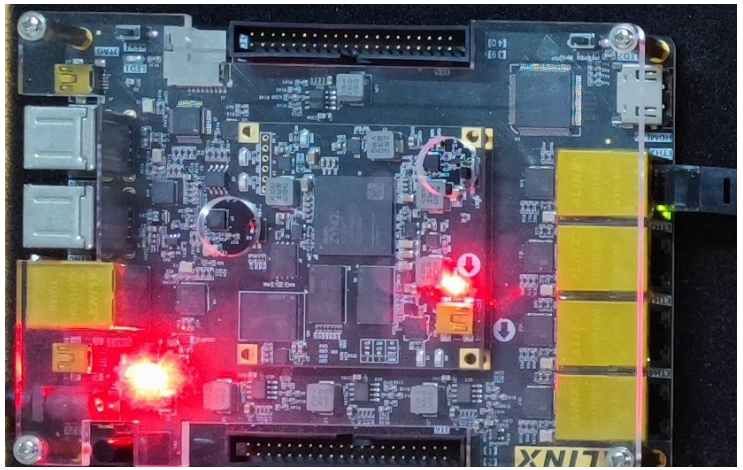
ZIGGO 系统特色

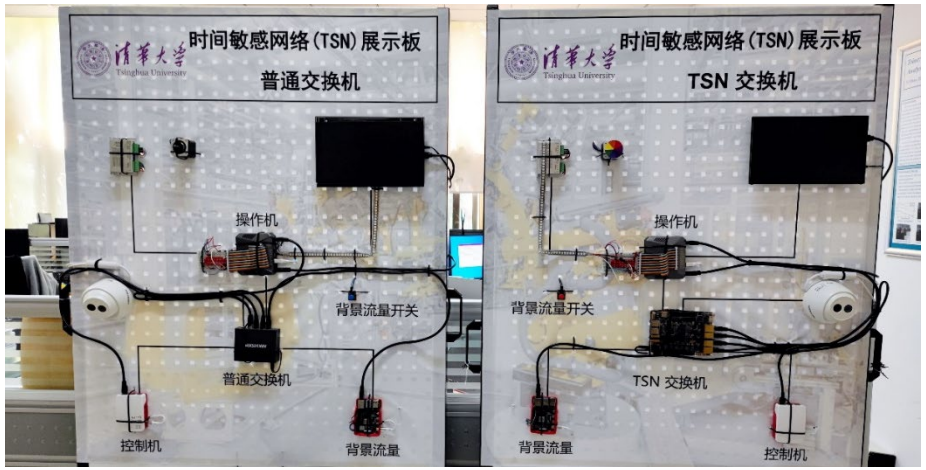
- 全面支持802.1AS、Qav、Qbv、Qcc等协议
- 支持IT流量与OT流量的共网传输
- 实现关键数据流量的确定性转发与超低时延传输
- 基于FPGA的软硬件协同设计
- TSN部分100%自主知识产权

纳秒 时间同步精度达到纳秒级

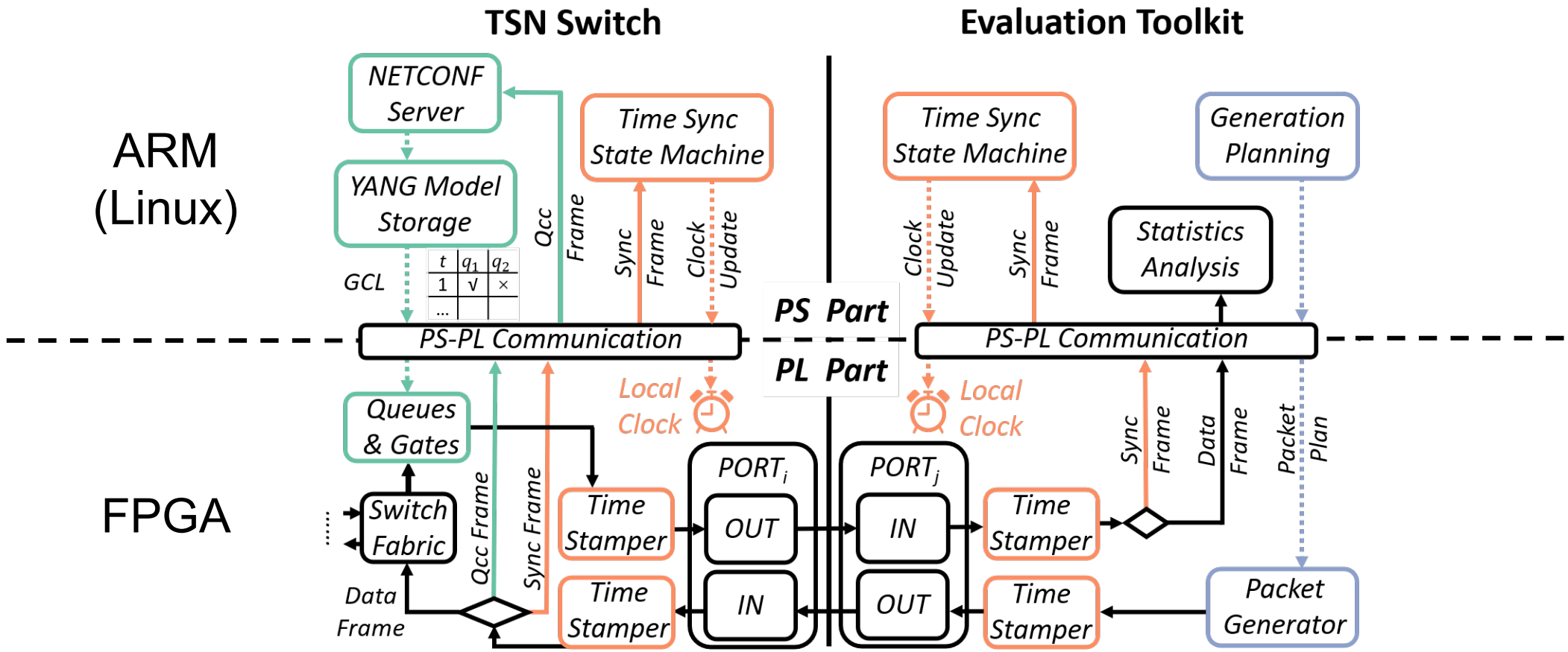
微秒 每跳时延抖动小于0.1微秒

万兆 带宽最大支持万兆以太网

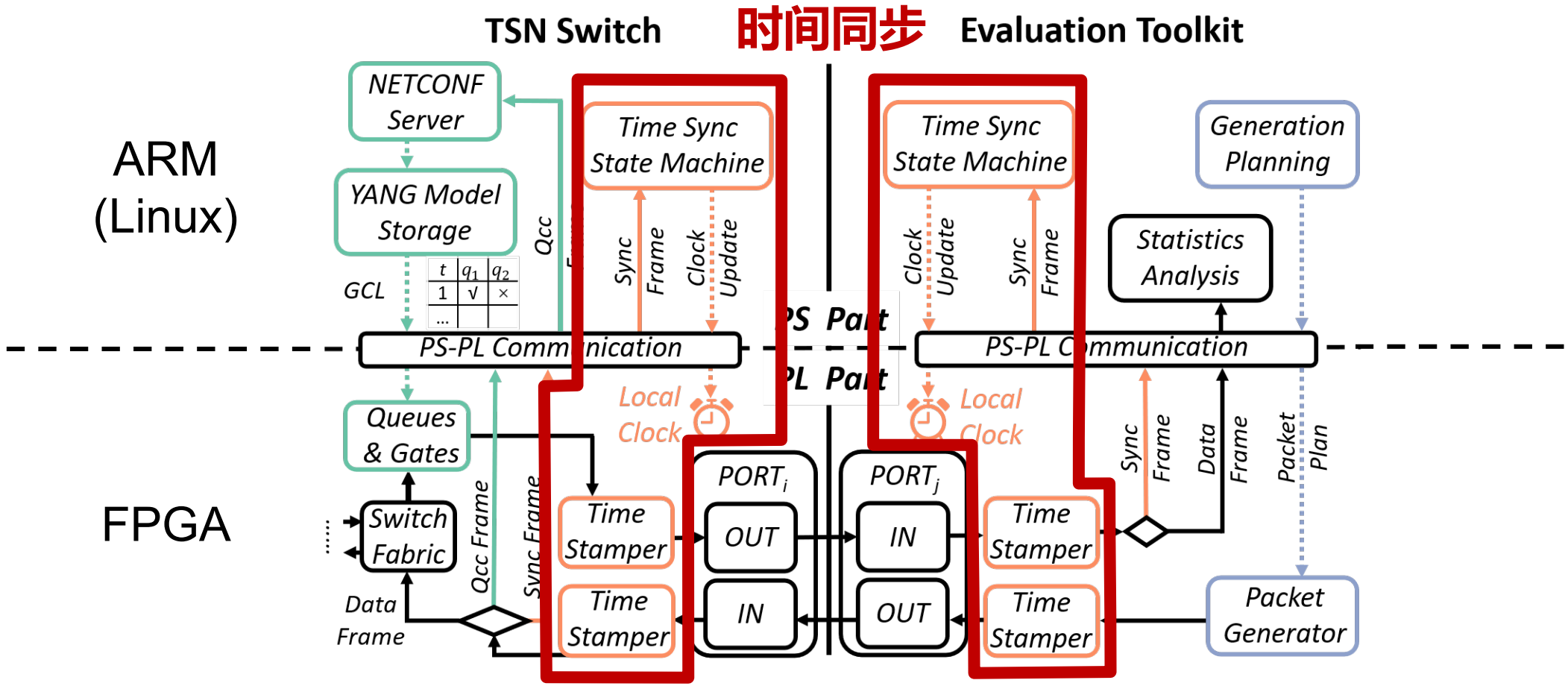




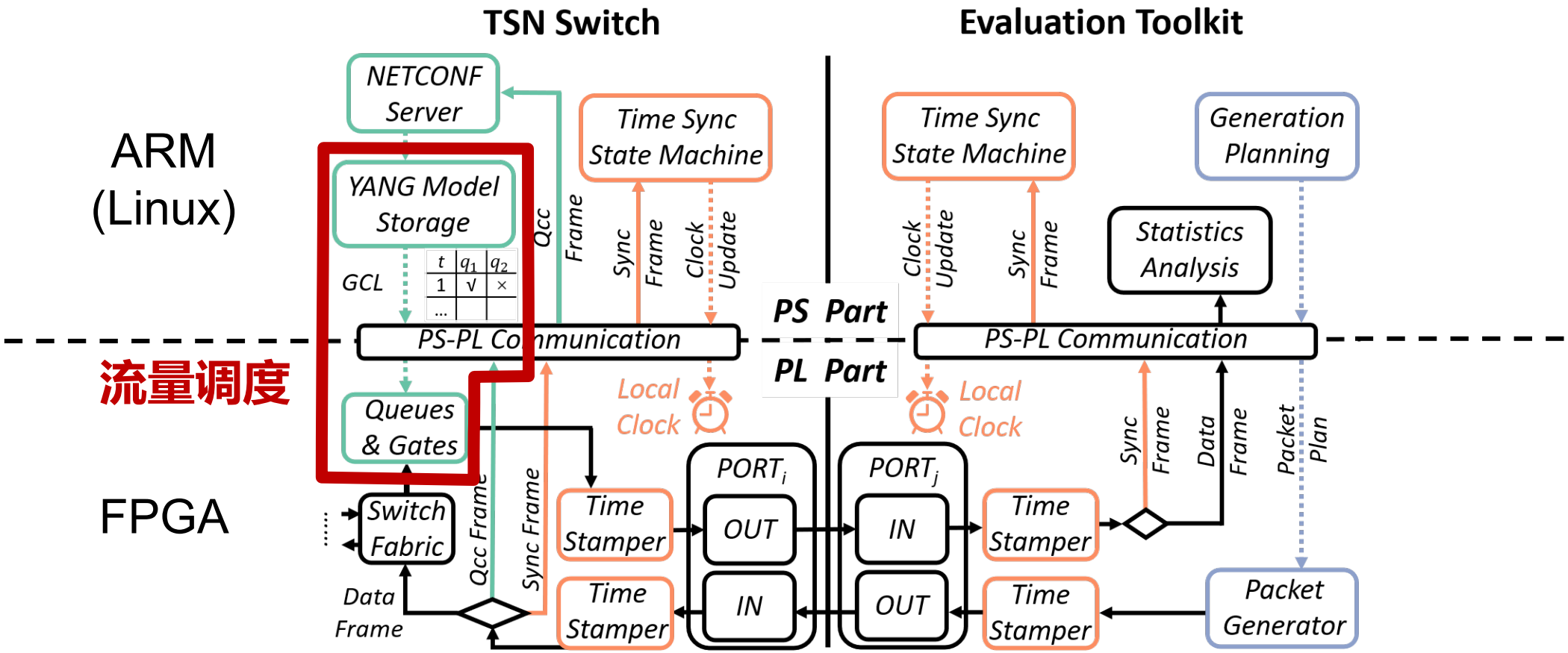
ZIGGO 架构设计


TSN Switch
Evaluation Toolkit
ARM
(Linux)
FPGA
NETCONF Server
YANG Model Storage
GCL
t | q1 | q2
1 | √ | ×
...
Qcc
Frame
Time Sync State Machine
Sync
Frame
Clock
Update
PS Part
PL Part
PS-PL Communication
Local Clock
Queues & Gates
Switch Fabric
Qcc Frame
Sync Frame
Data Frame
Time Stamper
PORT_i
OUT
IN
PORT_j
IN
OUT
Time Stamper
Time Stamper
Time Stamper
Time Sync State Machine
Clock
Update
Sync
Frame
PS-PL Communication
Local Clock
Statistics Analysis
Generation Planning
Sync
Frame
Data
Frame
Packet
Plan
Packet Generator

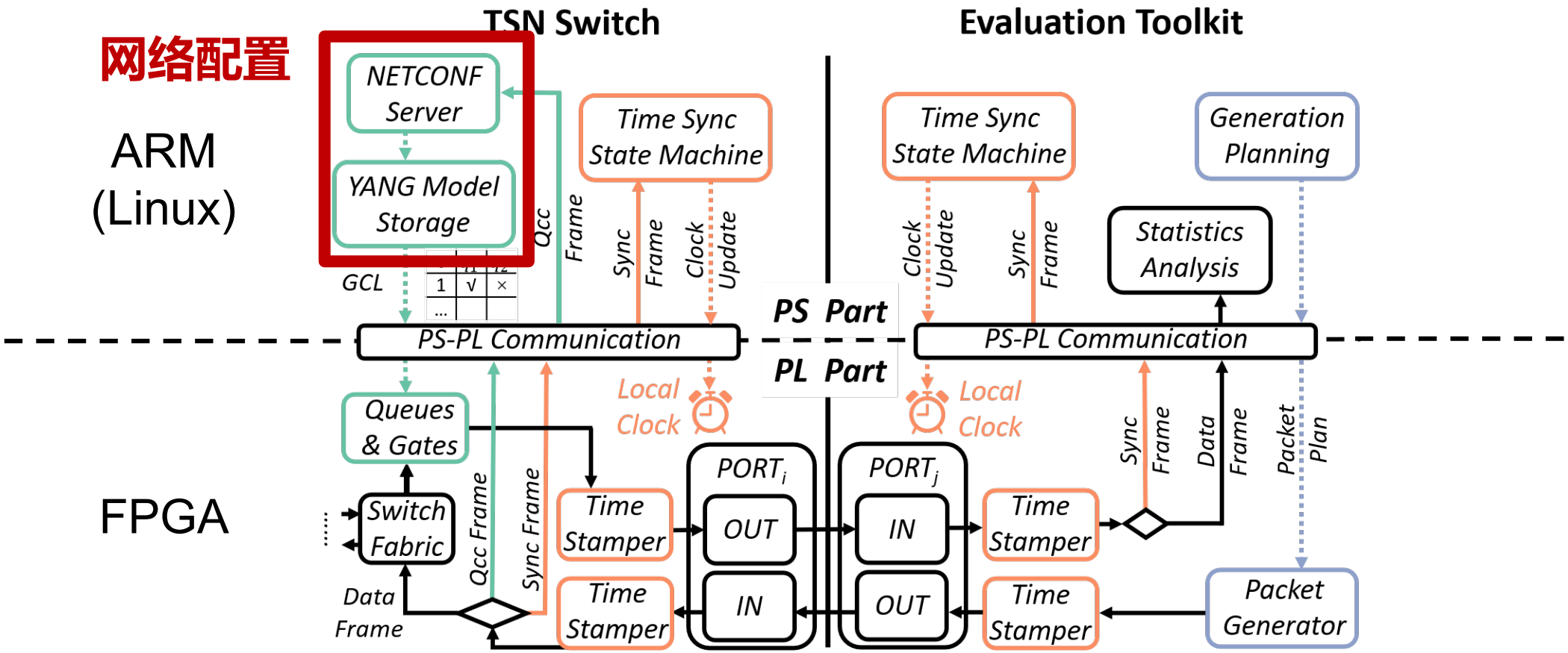
ZIGGO 架构设计


TSN Switch
时间同步
Evaluation Toolkit
ARM
(Linux)
FPGA
NETCONF Server
YANG Model Storage
GCL
Qcc
Time Sync State Machine
Sync
Frame
Clock
Update
PS Part
PL Part
PS-PL Communication
Local Clock
Queues & Gates
Switch Fabric
Data Frame
Qcc Frame
Sync Frame
Time Stamper
PORT$_i$
OUT
IN
PORT$_j$
Generation Planning
Statistics Analysis
Packet Plan
Packet Generator

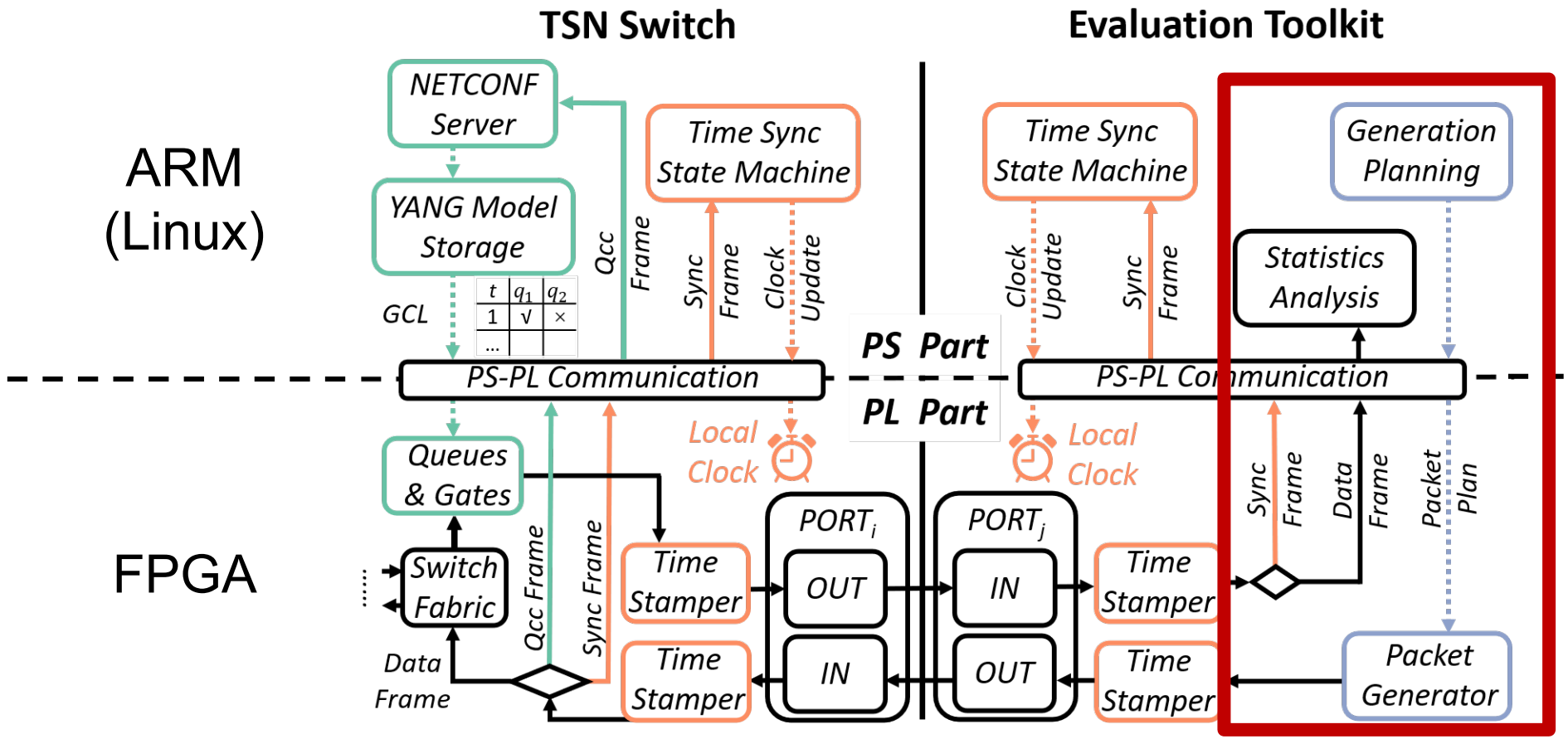
ZIGGO 架构设计


TSN Switch
Evaluation Toolkit
ARM
(Linux)
FPGA
流量调度
NETCONF Server
YANG Model Storage
GCL
t | q1 | q2
1 | √ | ×
...
Queues & Gates
Switch Fabric
Data Frame
Qcc
Frame
Qcc Frame
Sync Frame
Time Sync State Machine
Sync
Frame
Clock
Update
PS-PL Communication
Local Clock
PS Part
PL Part
PORTi
PORTj
OUT
IN
IN
OUT
Time Stamper
Time Stamper
Time Stamper
Time Stamper
Time Sync State Machine
Clock
Update
Sync
Frame
PS-PL Communication
Local Clock
Generation Planning
Statistics Analysis
Sync
Frame
Data
Frame
Packet
Plan
Packet Generator

ZIGGO 架构设计

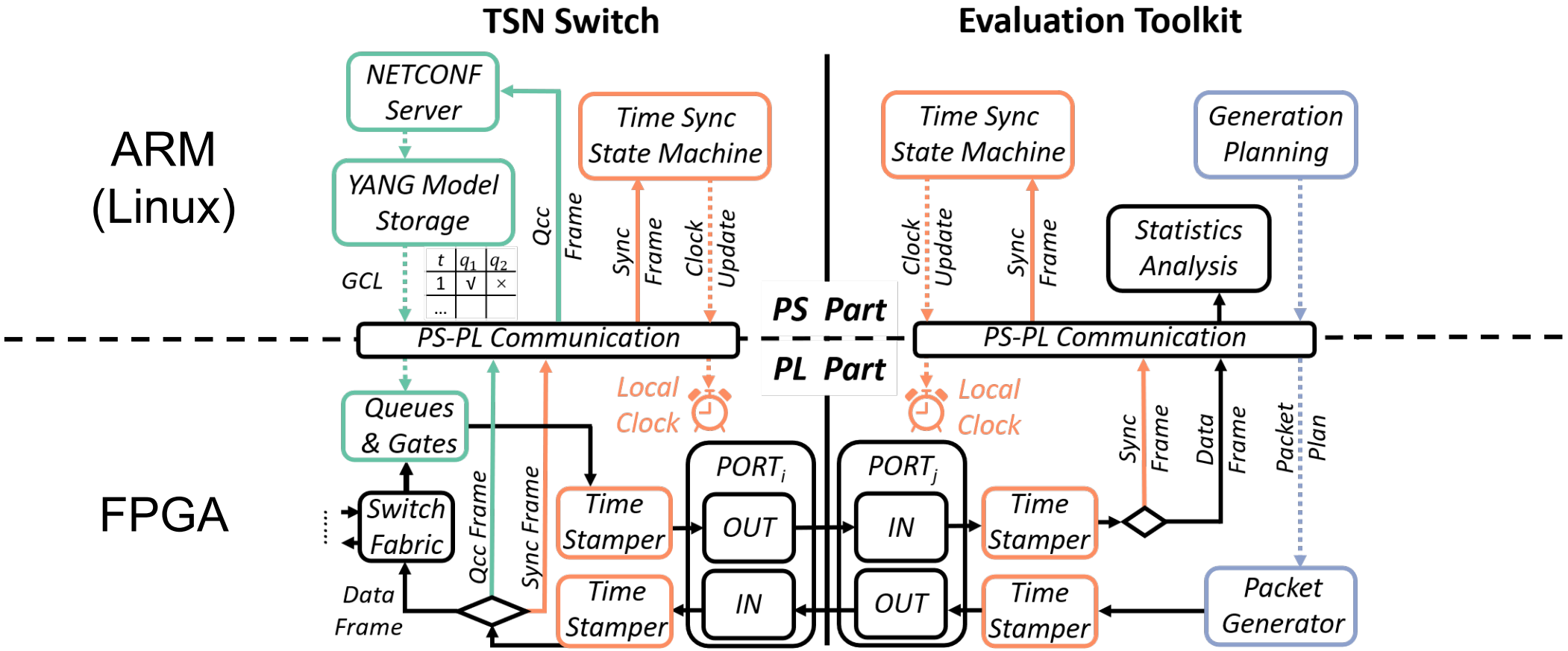

网络配置
ARM
(Linux)
FPGA
TSN Switch
Evaluation Toolkit
NETCONF Server
YANG Model Storage
GCL
Qcc
Frame
Time Sync State Machine
Sync
Frame
Clock
Update
PS Part
PL Part
PS-PL Communication
Local Clock
Queues & Gates
Switch Fabric
Data Frame
Qcc Frame
Sync Frame
Time Stamper
PORT$_i$
OUT
IN
PORT$_j$
Generation Planning
Statistics Analysis
Packet Plan
Packet Generator

ZIGGO 架构设计



网络探测与性能评估

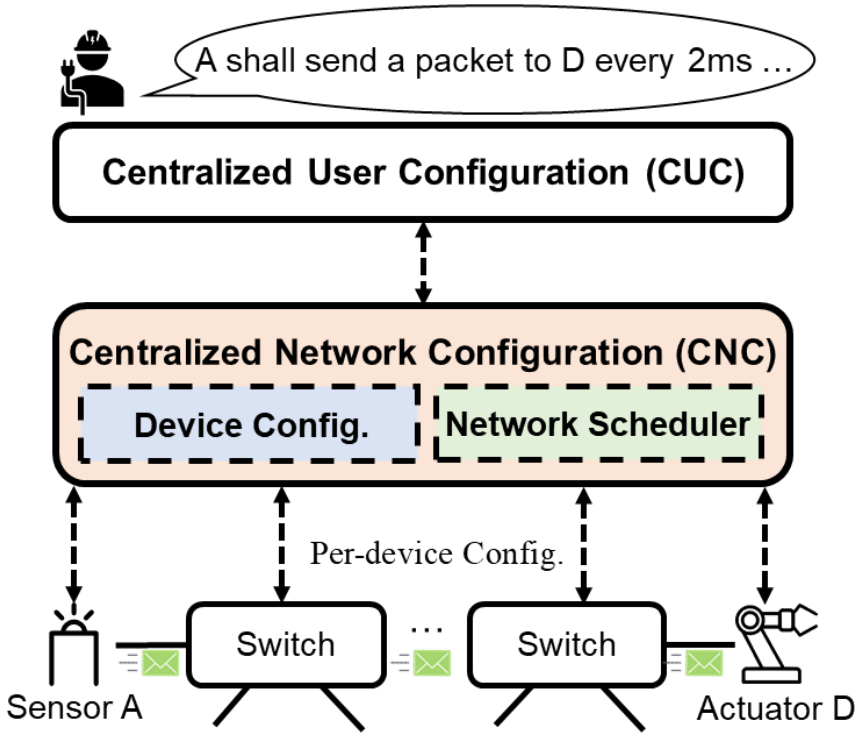
ZIGGO 架构设计



基于软硬件协同设计，实现确定性低延迟数据传输与分析

ZIGGO 集中配置

- 基于 IEEE 802.1Qcc 的 **CNC 集中网络配置模型**
- **TSN 调度器**根据用户需求求解关键流量传输时间
- **NETCONF/YANG 协议**从 CNC 向设备端发送配置



ZIGGO 开放平台

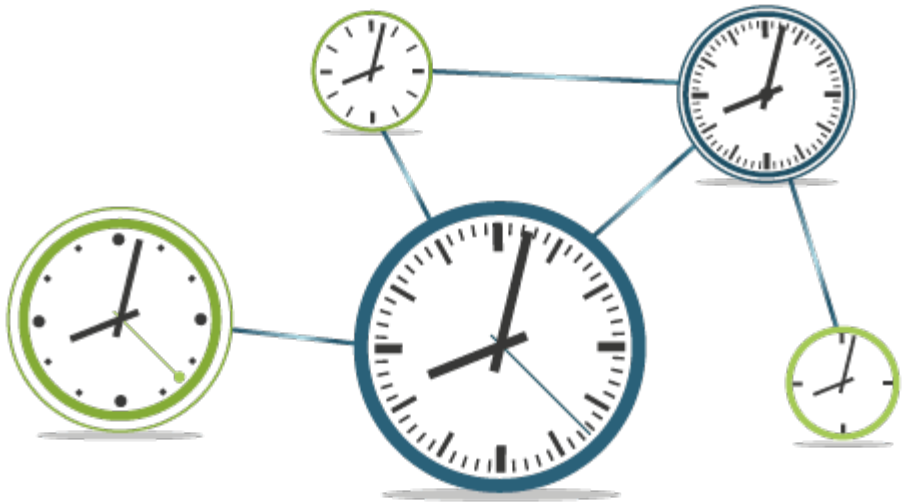
- 项目网站：http://tns.thss.tsinghua.edu.cn/ziggo/
- 开源工具：https://github.com/Horacehxw/Ziggo-Evaluation-Toolkit
  - 基于 Zynq7000 软硬件协同架构的即插即用 TSN 网络性能评估工具。
  - 支持 802.1AS 时间同步、符合 Qbv 标准的测试包收发与分析。

ABOUT ZIGGO

A *flexible*, *standard-compliant*, and *control-function-virtualized* TSN switch platform ready for *industrial control*, *automotive electronics*, and other time-sensitive applications.

ZIGGO TSN Switch 2.0

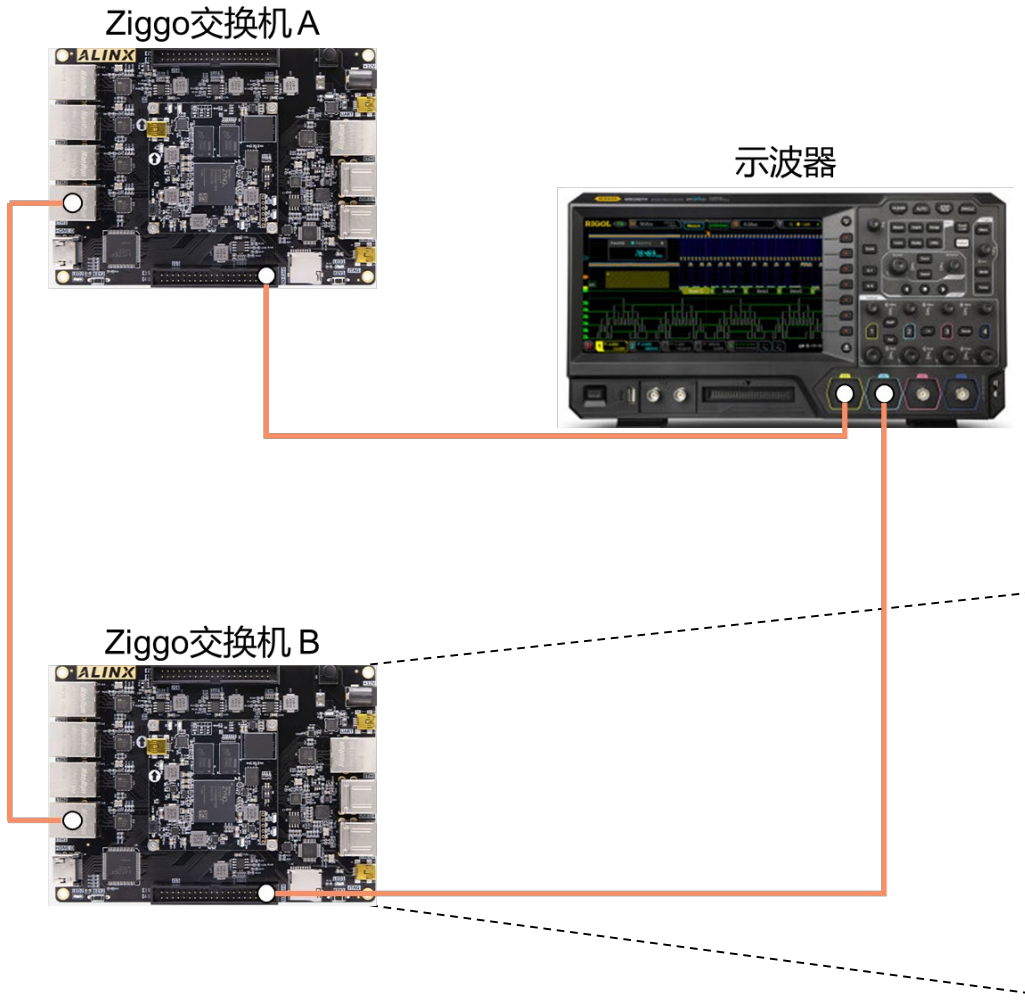
- **系统固件：**即插即用，适用于 AX7021 开发板的启动 SD 卡
  - 包含引导程序、硬件 Bitstream、设备树文件、操作系统、和根文件系统
- **软件代码：**
  - Verilog 代码，包含 TSN 交换机 FPGA 硬件部分
- **硬件代码：**
  - C/C++ 代码，包含时间同步状态机、CNC 配置接收端、PS-PL 通信模块
- **集中配置：**
  - Python/node.js 代码，包含流量调度、基于 JS 或 NETCONF 的网络配置
- **功能文档：**
  - 包括 Ziggo TSN Switch 2.0 在网络交换、时间同步、流量整形、集中配置等方面的功能和参数要求



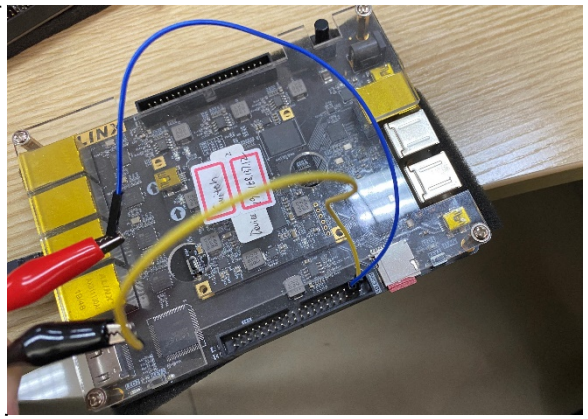
- Ziggo 交换机时间同步
- 互联互通时间同步
- 最优主时钟选择

时间同步：同一网络设备共享全局时钟

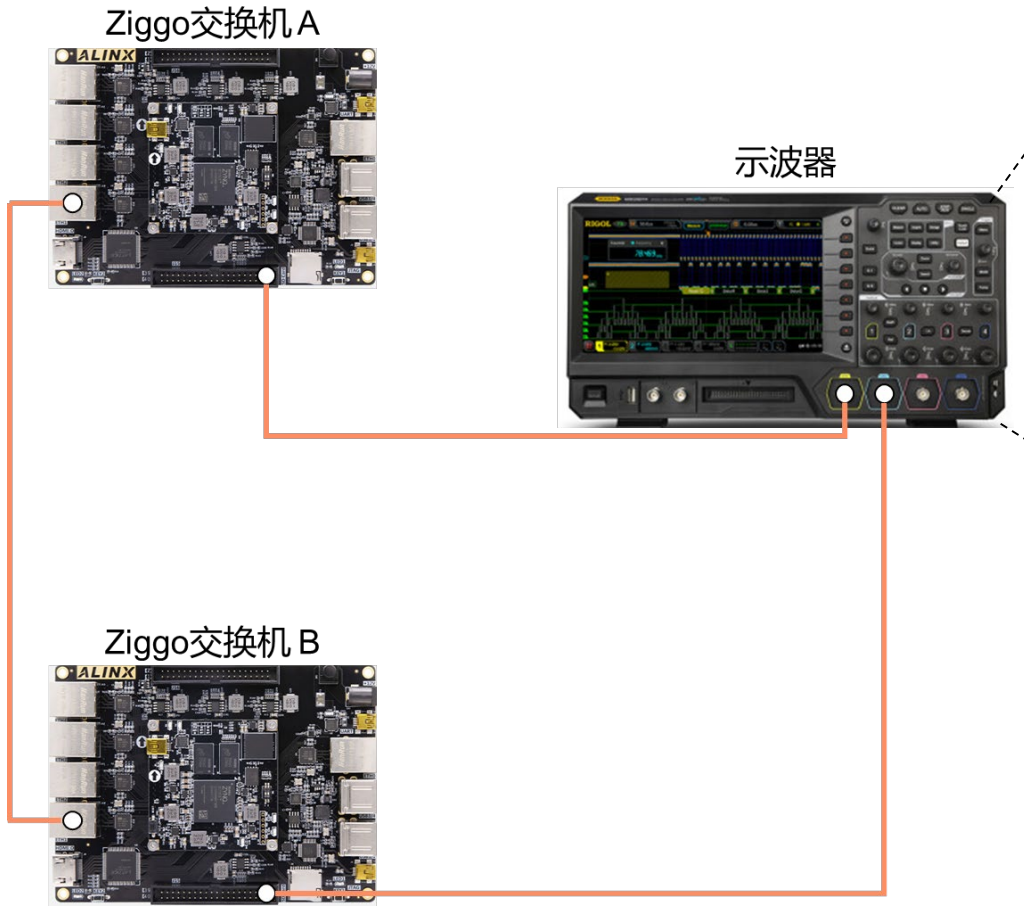
Ziggo 时间同步

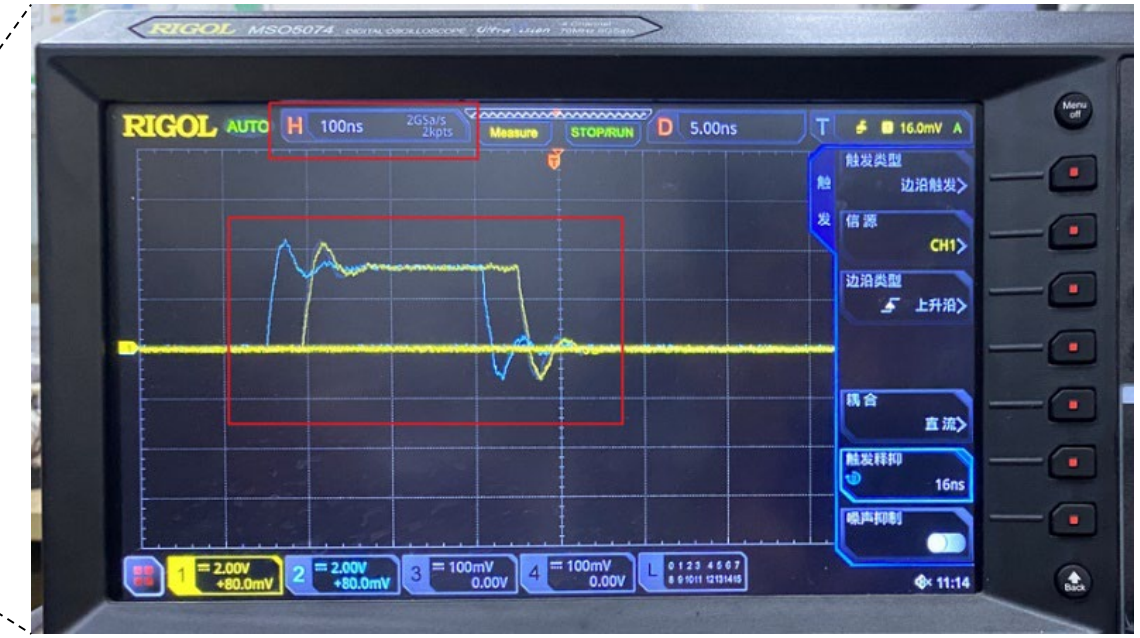


- 整数秒输出脉冲信号
- 脉冲信号长度 200ns
- 通过示波器分析同步误差



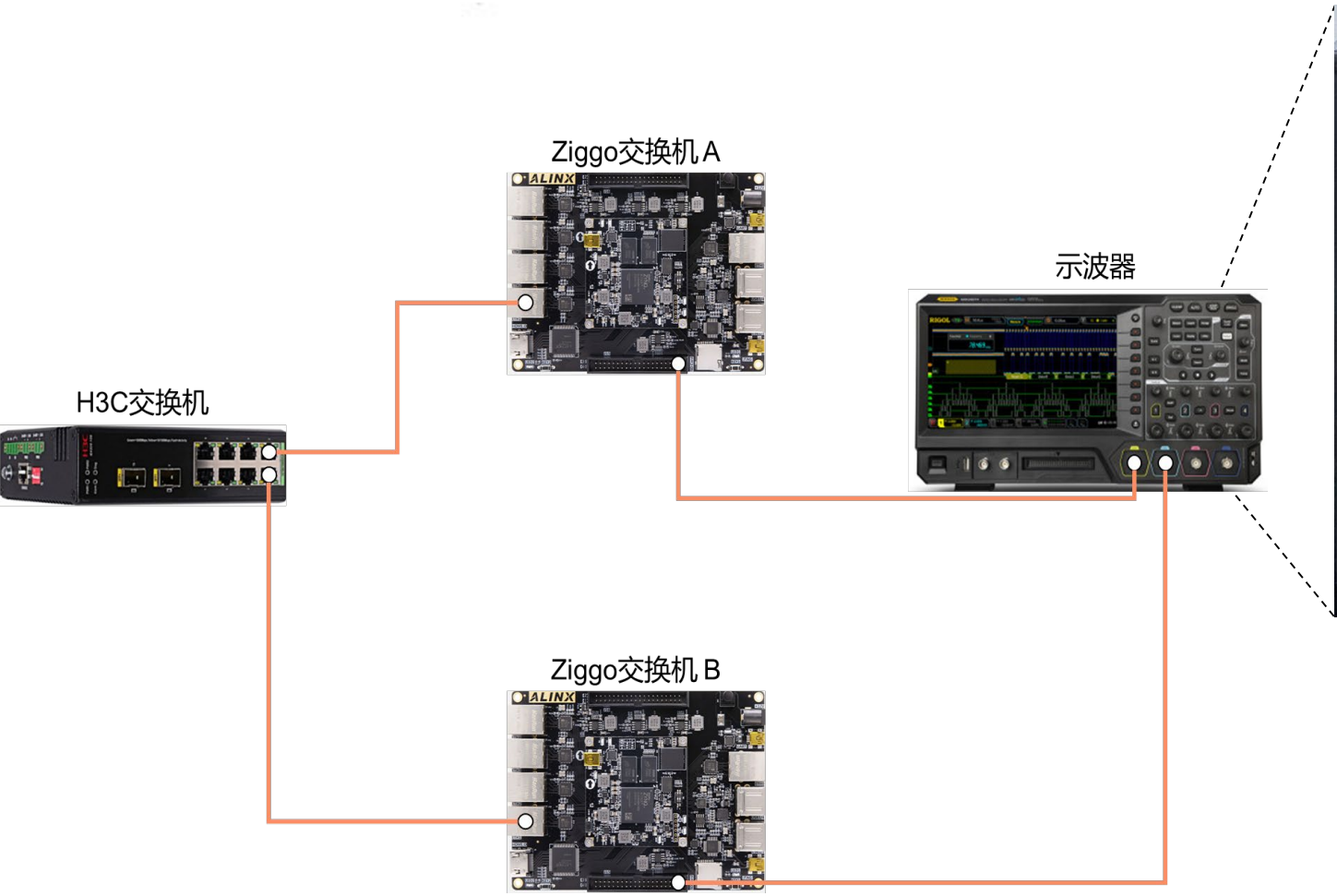
Ziggo 时间同步





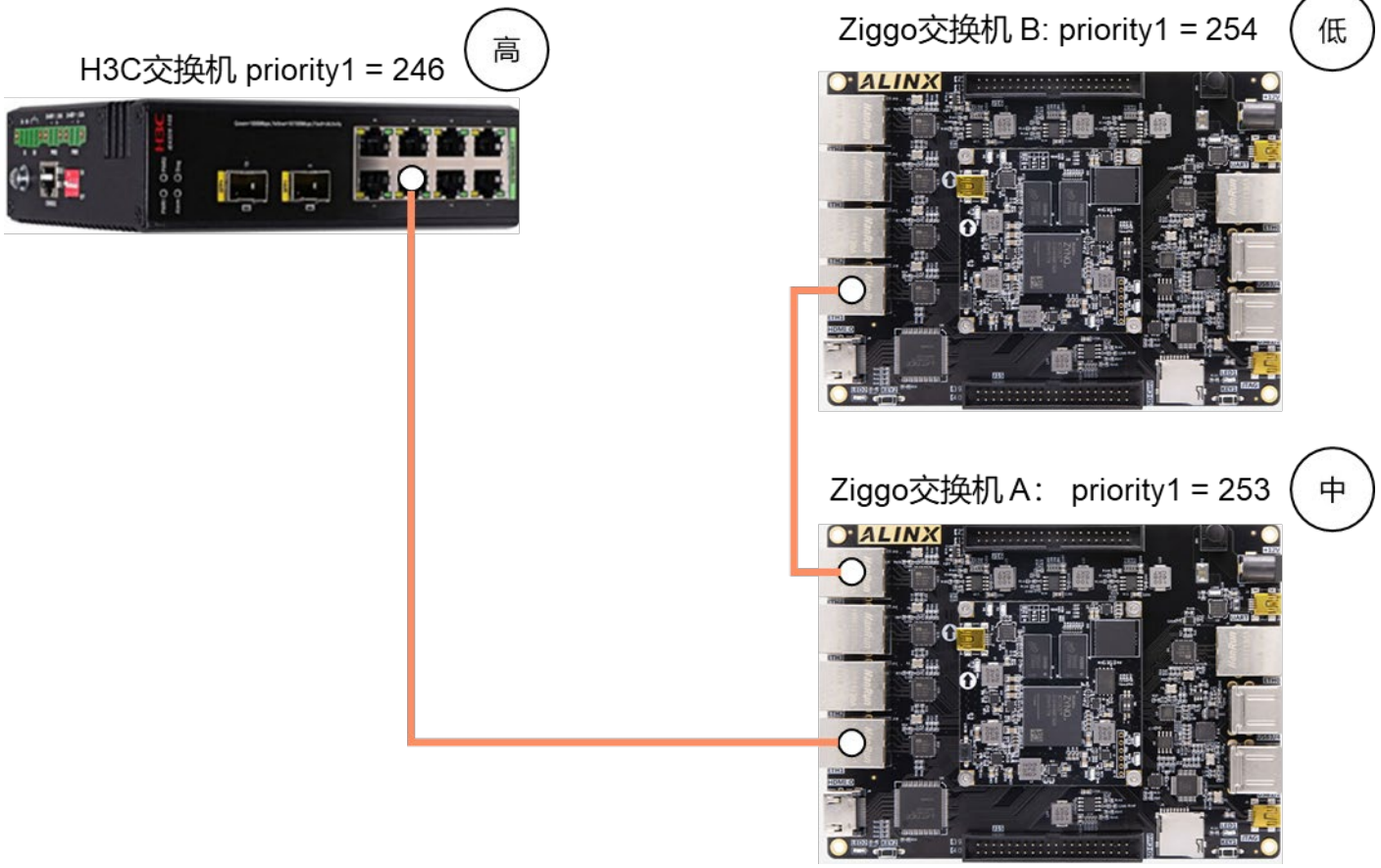
Ziggo同步误差
小于 100 纳秒

互联互通时间同步



支持不同品牌交换机时间同步协议互联互通

最优主时钟选择



- 交换机 B 依次连接 A 和 H3C 交换机, 观察 A/B 运行状态
  - 优先级：H3C> A > B
  - **结果：交换机 B, 交换机 A, H3C 依次成为主时钟**

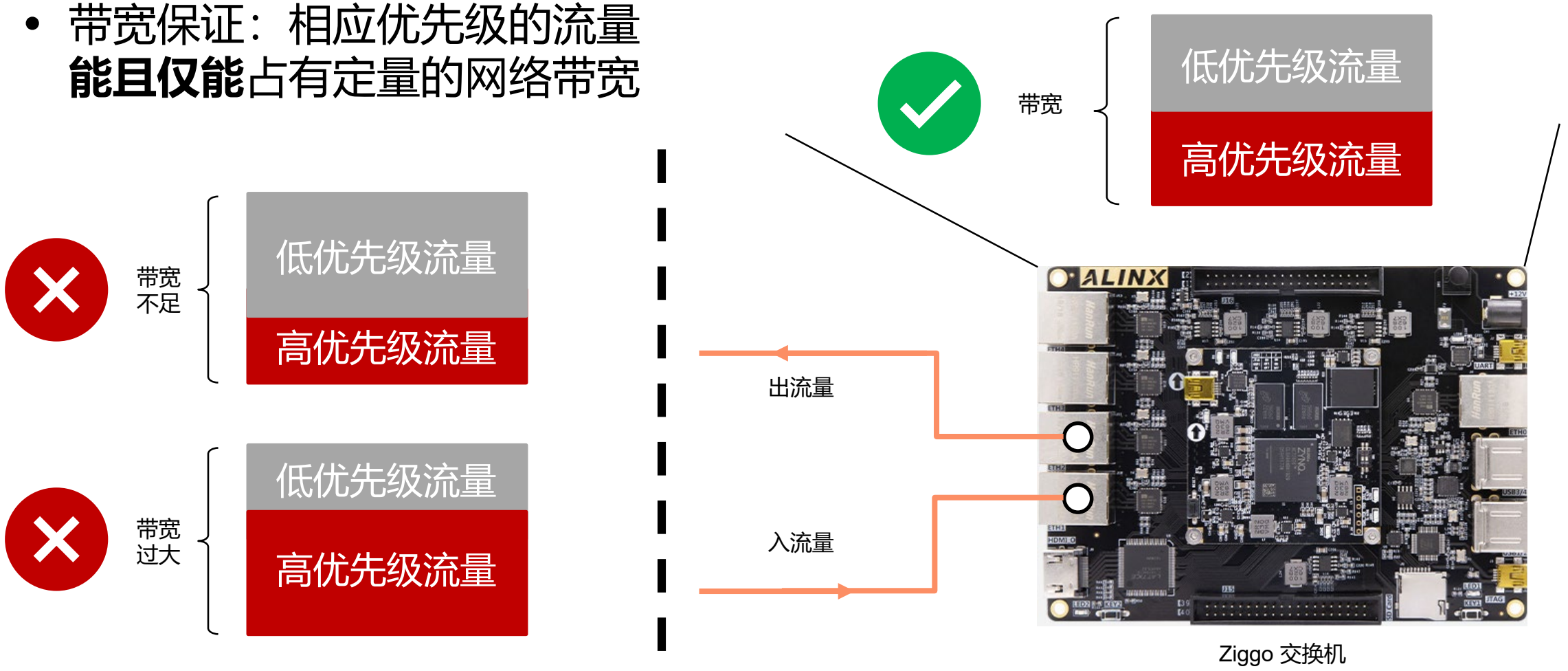
动态选择网络中最优节点作为主时钟

- Ziggo 交换机带宽保证

带宽保证：为高优先级流量预留足够带宽

带宽保证

- 带宽保证：相应优先级的流量**能且仅能**占有定量的网络带宽

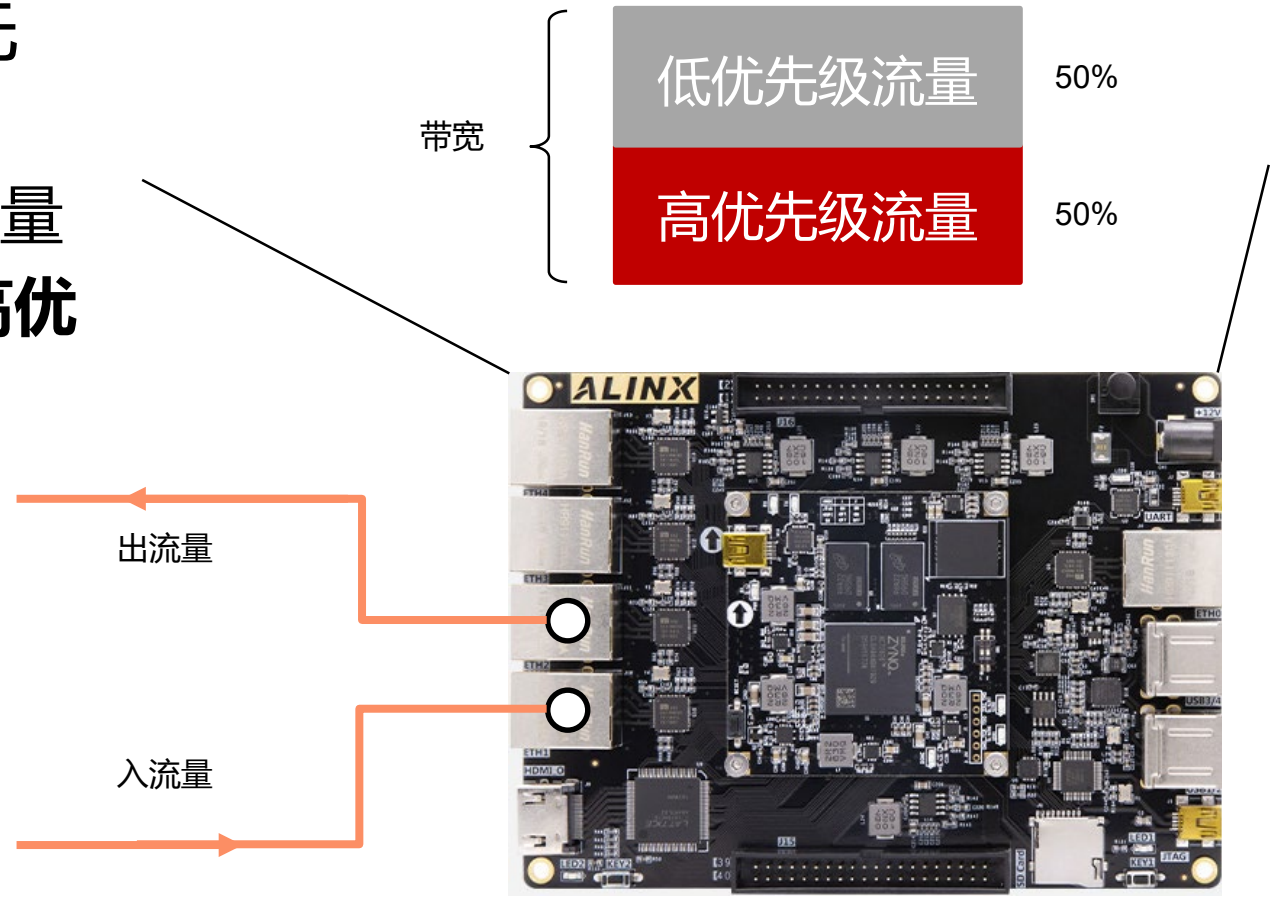


Ziggo 交换机

带宽保证

- 预留50%的带宽给高/低优先级流量
  - 打入50%/100%的高优先级流量
  - **结果：始终可以通过50%的高优先级流量**

Ziggo提供了精确的带宽保证



Ziggo 交换机

带宽保证

打入50%的高优先级流量，丢包率0%

```
(base) PS D:\Git\pcap_analyze> python .\analyse_packet.py .\50%_bandwidth_baseline.pcapng --step 1000
    Seq ID,  Received Number,  Max pkt_id,  Min pkt_id, Loss Rate,  Latency Mean,  Latency Variance
       257,             1000,         999,           0,      0.00,      58373.03,       20759279.28
       257,             1000,        1999,        1000,      0.00,      57521.75,       26626849.62
       257,             1000,        2999,        2000,      0.00,      55329.81,       27283075.65
       257,             1000,        3999,        3000,      0.00,      55045.32,       12393710.78
       257,             1000,        4999,        4000,      0.00,      58148.58,       20600249.55
       257,             1000,        5999,        5000,      0.00,      56817.82,       30423627.53
```

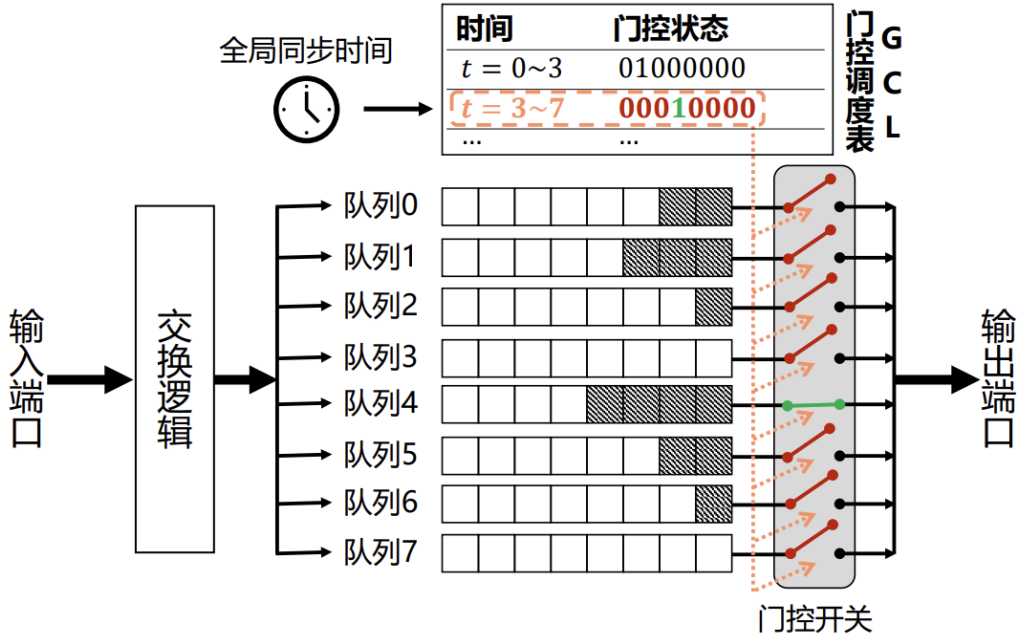
带宽保证

打入50%的高优先级流量，丢包率0%

```
(base) PS D:\Git\pcap_analyze> python .\analyse_packet.py .\50%_bandwidth_baseline.pcapng --step 1000
    Seq ID,  Received Number,  Max pkt_id,  Min pkt_id, Loss Rate,  Latency Mean,  Latency Variance
       257,             1000,         999,           0,      0.00,      58373.03,       20759279.28
       257,             1000,        1999,        1000,      0.00,      57521.75,       26626849.62
       257,             1000,        2999,        2000,      0.00,      55329.81,       27283075.65
       257,             1000,        3999,        3000,      0.00,      55045.32,       12393710.78
       257,             1000,        4999,        4000,      0.00,      58148.58,       20600249.55
       257,             1000,        5999,        5000,      0.00,      56817.82,       30423627.53
```

打入100%的高优先级流量，丢包率50%

```
(base) PS D:\Git\pcap_analyze> python .\analyse_packet.py .\100%_bandwidth.pcapng --step 2752
    Seq ID,  Received Number,  Max pkt_id,  Min pkt_id, Loss Rate,  Latency Mean,  Latency Variance
       257,             2752,        4115,           0,      0.33,     171104.74,  1015424371201.15
       257,             2752,        9607,        4116,      0.50,     298474.54,  2221382783736.31
       257,             2752,       15099,        9608,      0.50,     298464.63,  2221385428095.50
       257,             2752,       20591,       15100,      0.50,     298471.03,  2221384737446.77
       257,             2752,       26083,       20592,      0.50,     298465.10,  2221385328024.28
       257,             2752,       31575,       26084,      0.50,     298467.91,  2221385402732.33
```

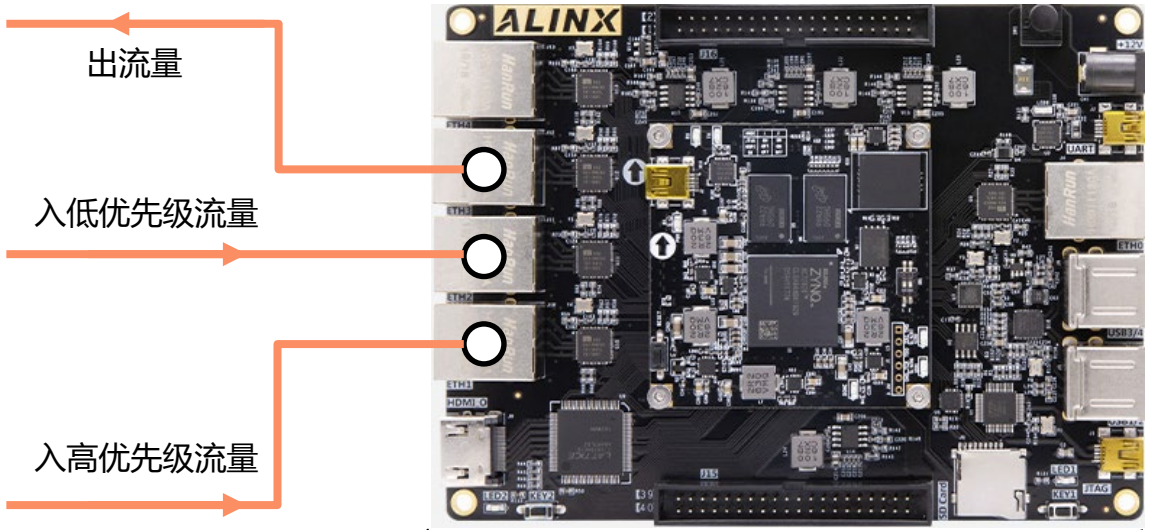


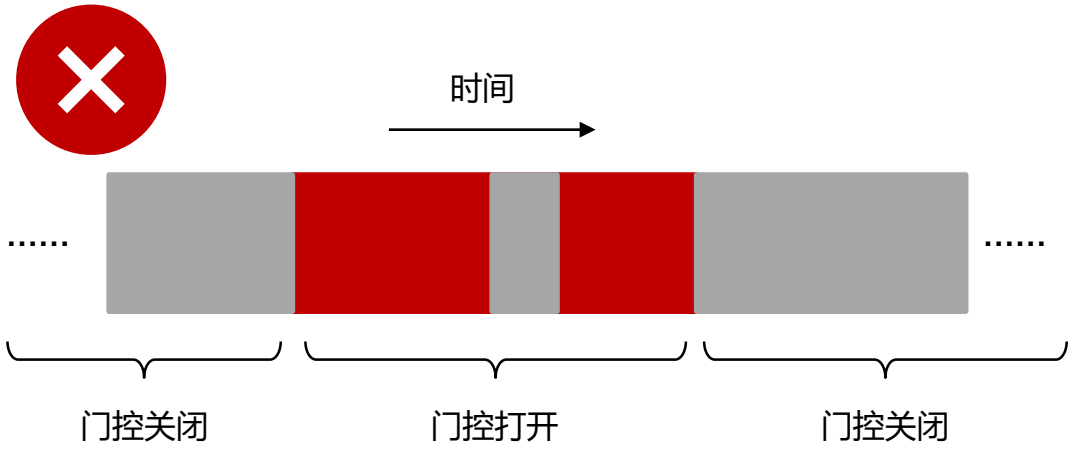
- Ziggo 交换机门控能力
- Ziggo 交换机门控精度

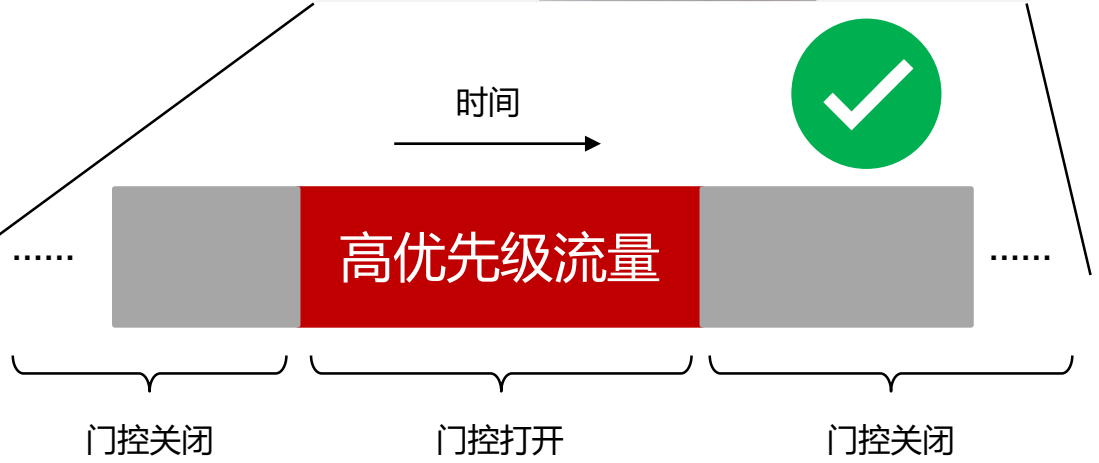
门控能力：支持特定流量在特定时隙传输

门控能力

- 门控能力：门控打开时，Ziggo交换机**仅会**转发高优先级数据包







门控能力

- 每个调度周期，门控打开 3 个时隙
  - 理论可以发出5个高优先级数据包
  - 测试：每个周期发送4/8个高优先级数据包
  - **结果：丢包率分别为 0%(0/4), 37.5%(3/8)**

Ziggo提供了正确的门控能力

发送4个包：丢包率0%(0/4)

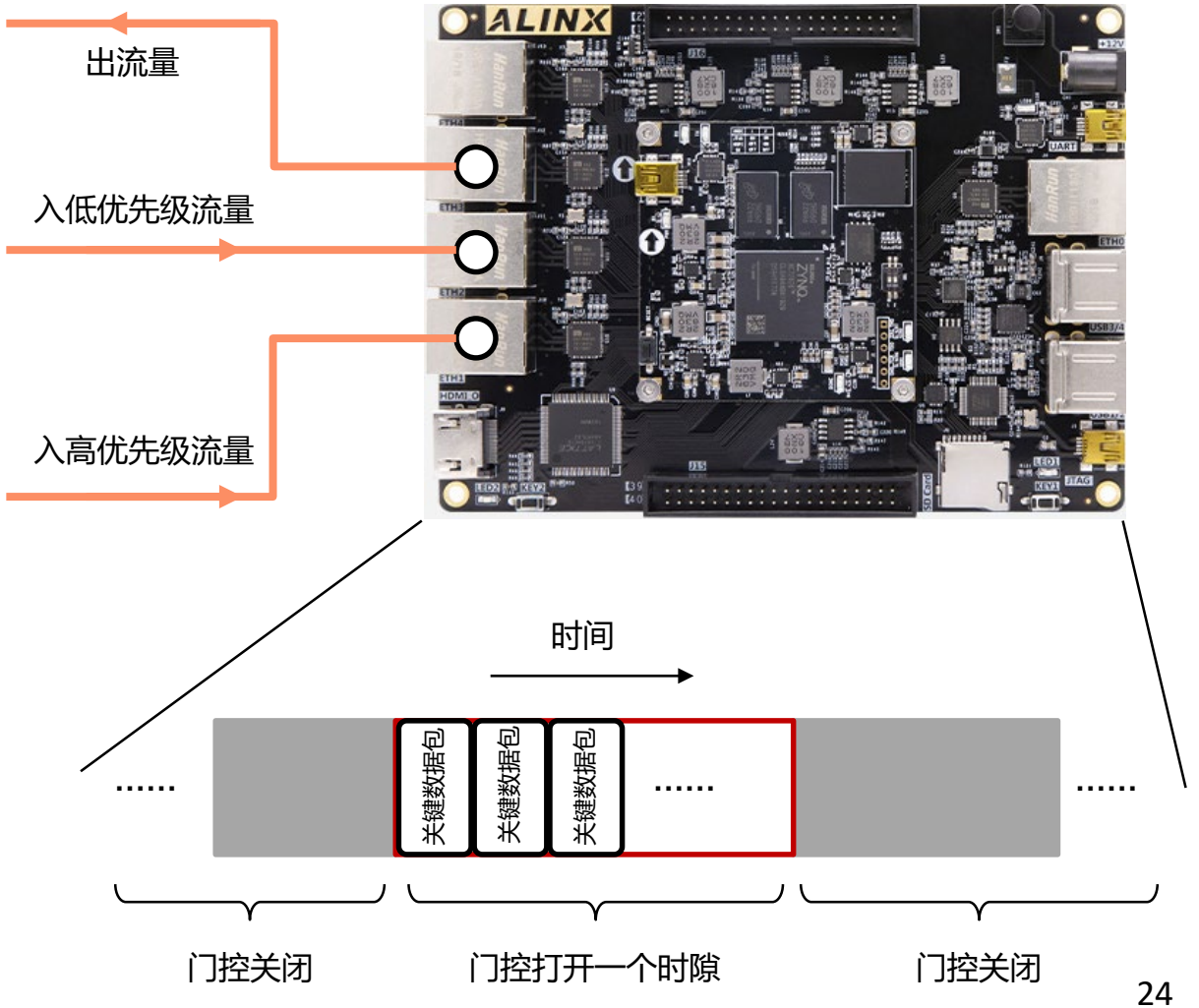
```
0101, 1000, 602000, 602999, 0.000000,  66641, 7012
0101, 1000, 603000, 603999, 0.000000,  66335, 7571
0101, 1000, 604000, 604999, 0.000000,  66150, 4316
0101, 1000, 605000, 605999, 0.000000,  66070, 1778
0101, 1000, 606000, 606999, 0.000000,  65979, 268770
0101, 1000, 607000, 607999, 0.000000,  65722, 154479
0101, 1000, 608000, 608999, 0.000000,  65836, 43043
0101, 1000, 609000, 609999, 0.000000,  66012, 2533
0101, 1000, 610000, 610999, 0.000000,  65970, 275041
```

发送8个包：丢包率37.5%(3/8)

```
0101, 625, 10002, 11001, 0.375000,  67162814, 2079
0101, 625, 11002, 12001, 0.375000,  67162809, 2139
0101, 625, 12002, 13001, 0.375000,  67162815, 1781
0101, 625, 13002, 14001, 0.375000,  67162825, 1396
0101, 625, 14002, 15001, 0.375000,  67162908, 31128
0101, 625, 15002, 16001, 0.375000,  67162794, 5140
0101, 625, 16002, 17001, 0.375000   67162811, 1261
0101, 625, 17002, 18001, 0.375000,  67162821, 1747
```

门控精度

- 门控精度：门控打开时长的精度
- 借助可转发**关键数据包的数量**衡量门控打开的时长

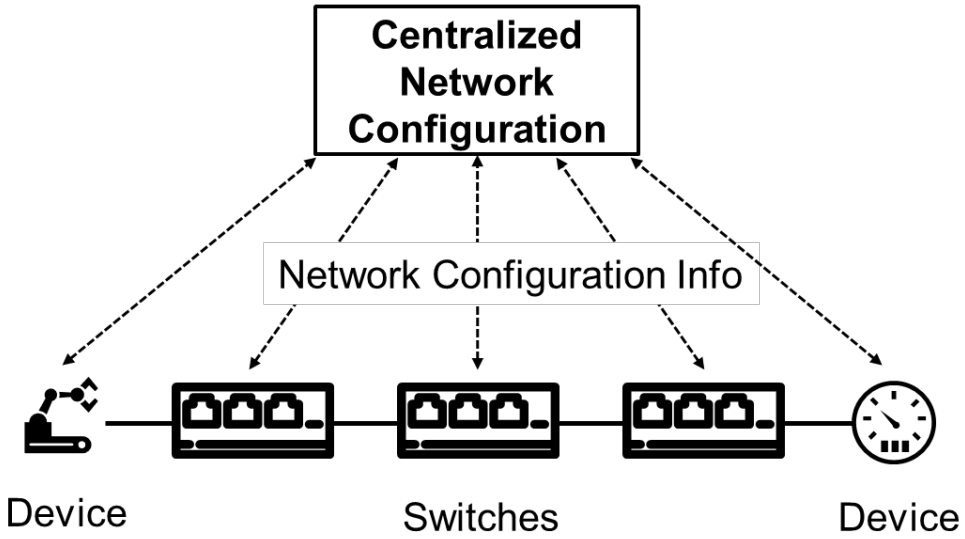


门控精度

- 以太网最小数据包大小为64Byte
- 理论一个时隙可以发送**32**个大小为64Byte的关键数据包
- **实验：Ziggo交换机恰好可以转发32个关键数据包，零丢包**

Ziggo 实现了精确的门控时长

```
device 44 latency:
Seq     ID,     Received        Number, Max     pkt_id, Min     pkt_id, Loss    Rate,   Latenc
y       Mean,   Latency Variance
0101,   1000,   22000,  22999,  0.000000,              65954,  45424
0101,   1000,   23000,  23999,  0.000000,              65895,  38431
0101,   1000,   24000,  24999,  0.000000,              65949,  22808
0101,   1000,   25000,  25999,  0.000000,              65910,  21643
0101,   1000,   26000,  26999,  0.000000,              65904,  22790
```



- TSN 流量调度
- 基于 JS 脚本的网络配置
- 基于 NETCONF/YANG 的网络配置

集中网络配置：通过中央控制器配置网络设备

TSN 流量调度

- 基于 **SMT 建模**，求解满足用户需求的的关键流量传输时间表

数据帧重叠约束

转发时延约束

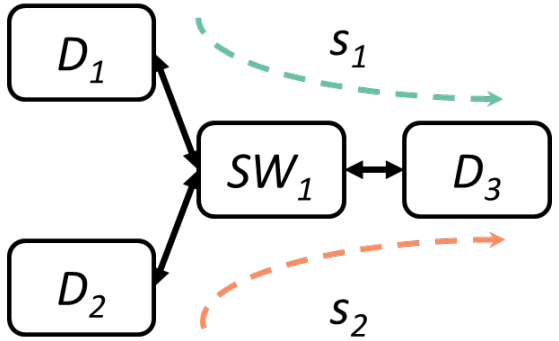
流量延迟约束

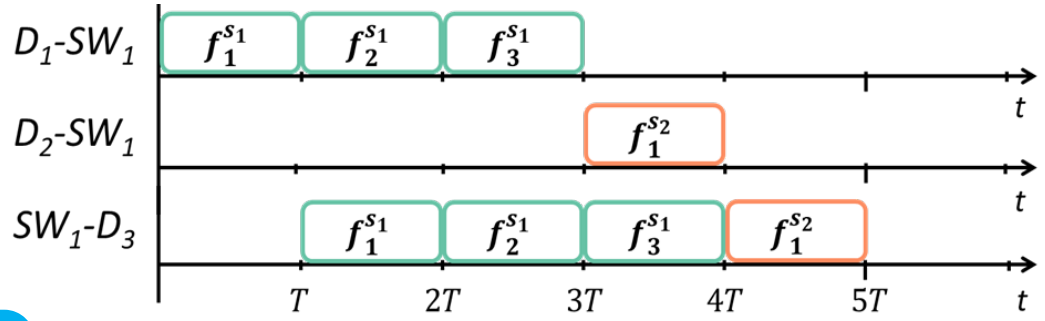
相邻链路约束

$$\forall j_i \in J, \forall f_{ik} \in F_i, \forall h \in V_{sw}:$$
$$m \leftarrow \begin{cases} f_{ik}.src, & \text{if } f_{ik} \text{ is an input flow,} \\ f_{ik}.dst, & \text{if } f_{ik} \text{ is an output flow,} \end{cases}$$
$$sp \leftarrow shortestpath(h, m).$$
$$\forall (s,e) \in E,$$
$$AddConstraints(Imply(j_i.host = h \wedge (s,e) \in sp, x_{ikse} = 1)),$$
$$AddConstraints(Imply(j_i.host = h \wedge (s,e) \notin sp, x_{ikse} = 0)),$$

$$\forall j_i \in J, \forall f_{ik} \in F_i:$$
$$\forall (s,e,c) \in \{(s,e,c) \in V \times V \times V | (s,e) \in E \wedge (e,c) \in E\}:$$
$$AddConstraints(Imply(x_{ikse} = 1 \wedge x_{ikec} = 1, (t_{ikec} + o_{ikec} \times j_i.P) - (t_{ikse} + o_{ikse} \times j_i.P) \geq e.d)$$

$$\forall j_a \in J, j_b \in J, a \neq b:$$
$$P^s \leftarrow lcm(j_a.P, j_b.P)$$
$$\forall u \in \{u \in \mathbb{N} | u \leq \frac{P^s}{j_a.P}\}, \forall v \in \{v \in \mathbb{N} | v \leq \frac{P^s}{j_b.P}\}:$$
$$AddConstraints(Imply(j_a.host = j_b.host, j_a.start + u \times j_a.P + j_a.T < j_b.start + v \times j_b.P \vee j_b.start + v \times j_b.P + j_b.T < j_a.start + u \times j_a.P))$$





Ziggo 实现了自动化流量调度

基于 JS 脚本的网络配置

- 通过 **JS 脚本 + SSH 指令**控制网络设备
  - distribute: 下发配置文件到所有设备
  - collect: 收集 device 接收到的数据包信息
  - launch: 按要求启动 TSN Switch 和 Device 程序
  - pull/build: 向各设备分发和编译软件代码

支持快速调试、错误定位、和代码修改

```
[
    {
        "job_id": 0, // CaaS中的任务ID，若不需要可以删掉
        "flow_id": 0, // 数据流ID
        "src": 1, // 数据流的源节点ID，Device类型，和拓扑文件中的ID对应
        "dst": 2, // 数据流的目的节点ID，Device类型，和拓扑文件中的ID对应
        "period": 2048, // 数据流的发送周期，单位是2^14ns
        "MD": 1024 // 最大允许时延，单位是2^14ns
    },
    {
        "job_id": 0,
        "flow_id": 1,
        "src": 1,
        "dst": 2,
        "period": 1024,
        "MD": 1024
    },
    {
        "job_id": 0,
        "flow_id": 2,
        "src": 2,
        "dst": 1,
        "period": 1024,
        "MD": 1024
    }
]
```

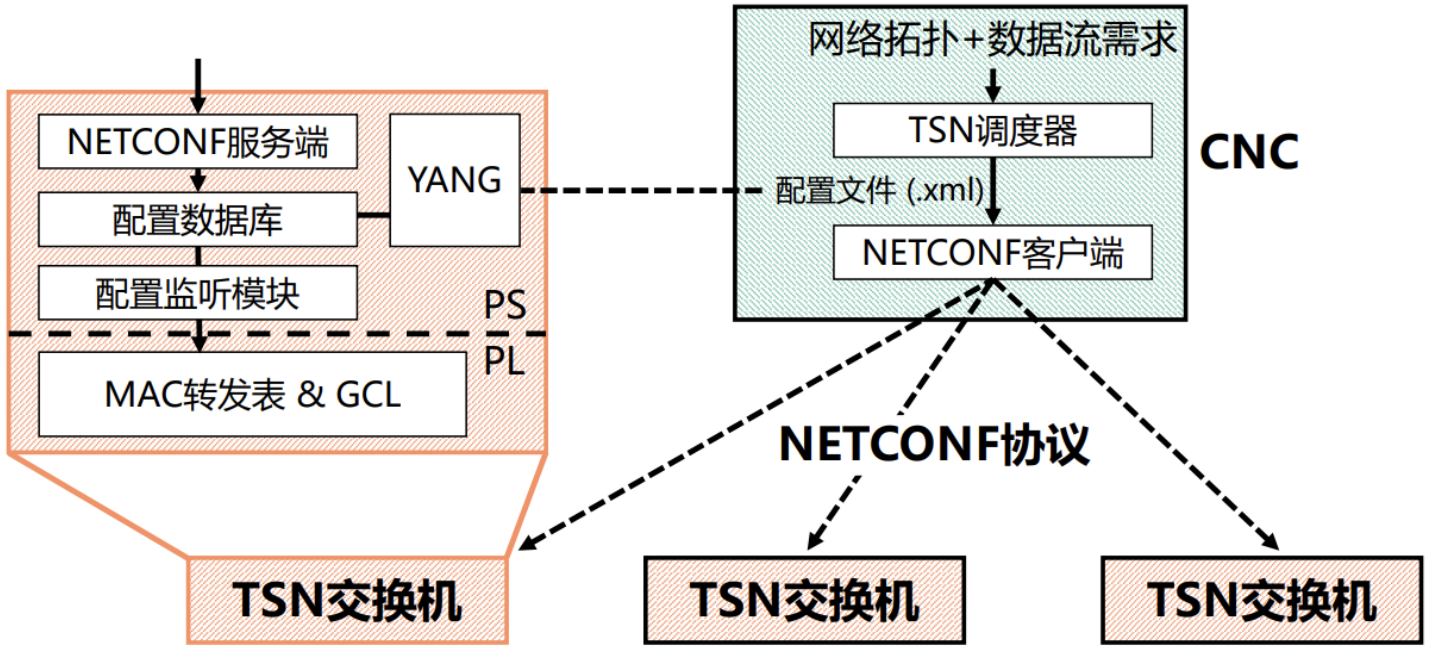
schedule.json: 流量时间表配置

```
{
    "nodes": [ // 用于描述网络中每个节点信息
        {
            "id": 0,  // 节点ID，和后面的"src"和"dst"
            "type": "switch",  // 节点类型，分为"swit
            "mac": "00:00:00:00:02:01", // 节点的物理
            "ptp_ports": [
                1,
                0,
                0,
                0,
                3
            ] // 节点的PTP端口状态
            // 五元组，分别代表 [local,ETH1,ETH2,ETH3,
            // 数字含义：(0: MASTER, 1: SLAVE, 2: PAS
            // 若local为1，代表该节点是主时钟节点；
            // 若local为0，代表该节点是从时钟节点。
        },
        {
            "id": 1,
            "type": "device",
            "mac": "00:00:00:00:00:01"
        }
    ],
    "links": [ // 拓扑信息，由每条link组成，每个link是一个有向边
        {
            "id": 0, // link ID
            "src": 0, // link的源节点ID
            "src_port": 0, // 源节点对应的端口号
                           // port 0,1,2,3对应实际中的ETH1,2,3,4
            "dst": 1, // link的目的节点ID
            "dst_port": 0 // 目的节点对应的端口号
        },
        {
            "id": 1,
            "src": 0,
            "src_port": 1,
            "dst": 2,
            "dst_port": 0
```

config.json: 网络拓扑和主从时钟配置

基于 NETCONF/YANG 的网络配置

- 基于 NETCONF/YANG 配置交换机的**GCL和MAC转发表**
- 根据 **NETCONF 协议**，从 CNC 向每个网络设备发送配置信息
- 交换机在配置数据库中维护 **YANG 模型**定义的配置信息，实时**监听配置变动**



Ziggo 完全支持 IEEE 802.1Qcc 集中配置

谢谢聆听