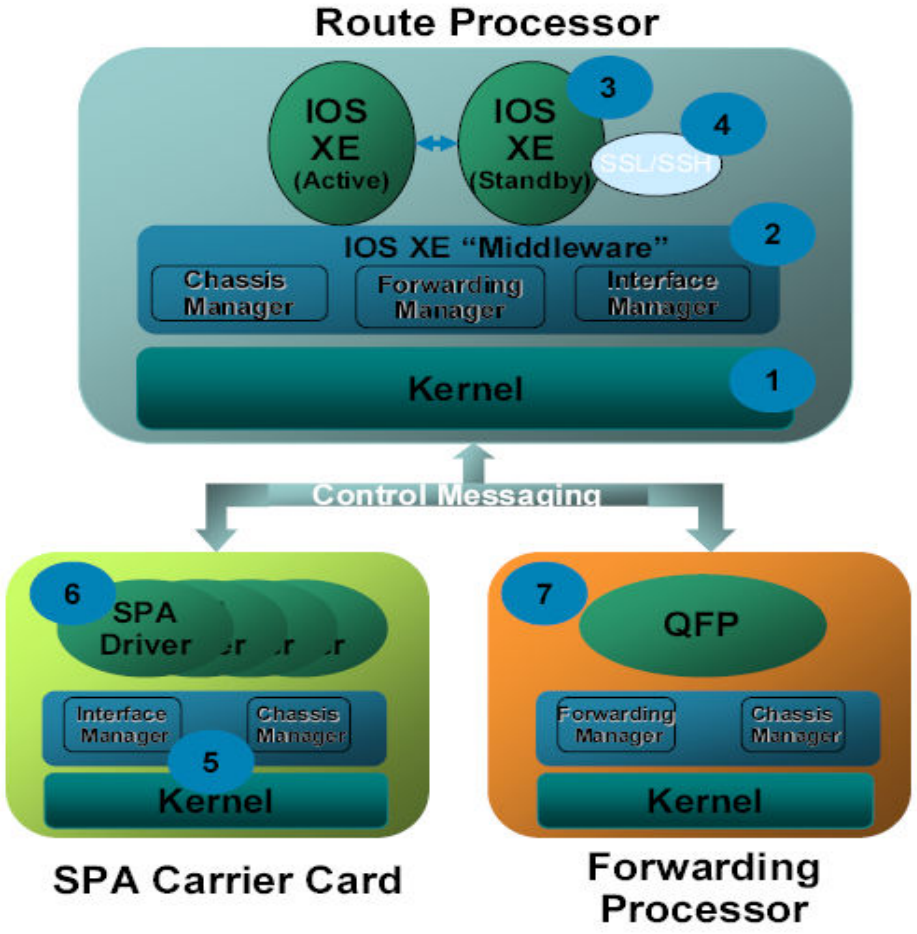
ASR 1000 Software Components & Packaging

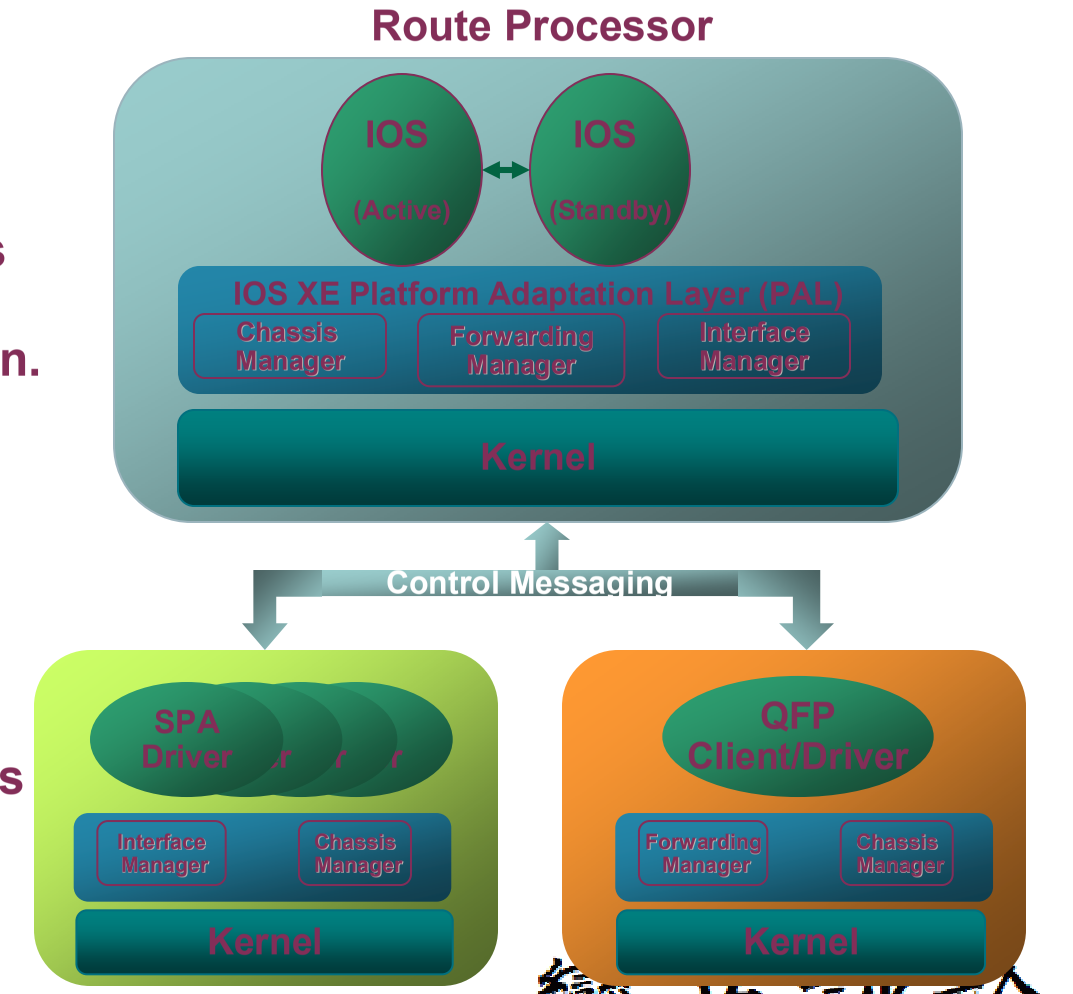
1. **RP-Base** (RP OS)
2. **RP-Control** (Control Plane processes that interface between IOS and the rest of the platform)
3. **RP-IOS**
4. **RP-Access (K9 & non-K9)** (Software required for Router access – SSH, SSL)
5. **SIP-Base** (SIP OS + Control processes)
6. **SIP-SPA** (SPA drivers and FPD (SPA FPGA image))
7. **ESP-Base** (ESP OS + Control processes + QFP client/driver/ucode)

+ ROM Monitor: One ROM Monitor package containing ROMMON for RP, ESP, SIP



Software Architecture – IOS XE

- IOS XE = IOS + IOS XE Middleware + Platform Software
- Operational Consistency - same look and feel as IOS Router
- IOS runs as its own Linux process for control plane (Routing, SNMP, CLI etc). Capable of 64bit operation.
- Linux kernel with multiple processes running in protected memory for
  - Fault containment
  - Re-stability
  - ISSU of individual SW packages
- ASR 1000 HA Innovations
  - Zero-packet-loss RP Failover
  - <50ms ESP Failover
  - "Software Redundancy"



Software Architecture – IOS XE

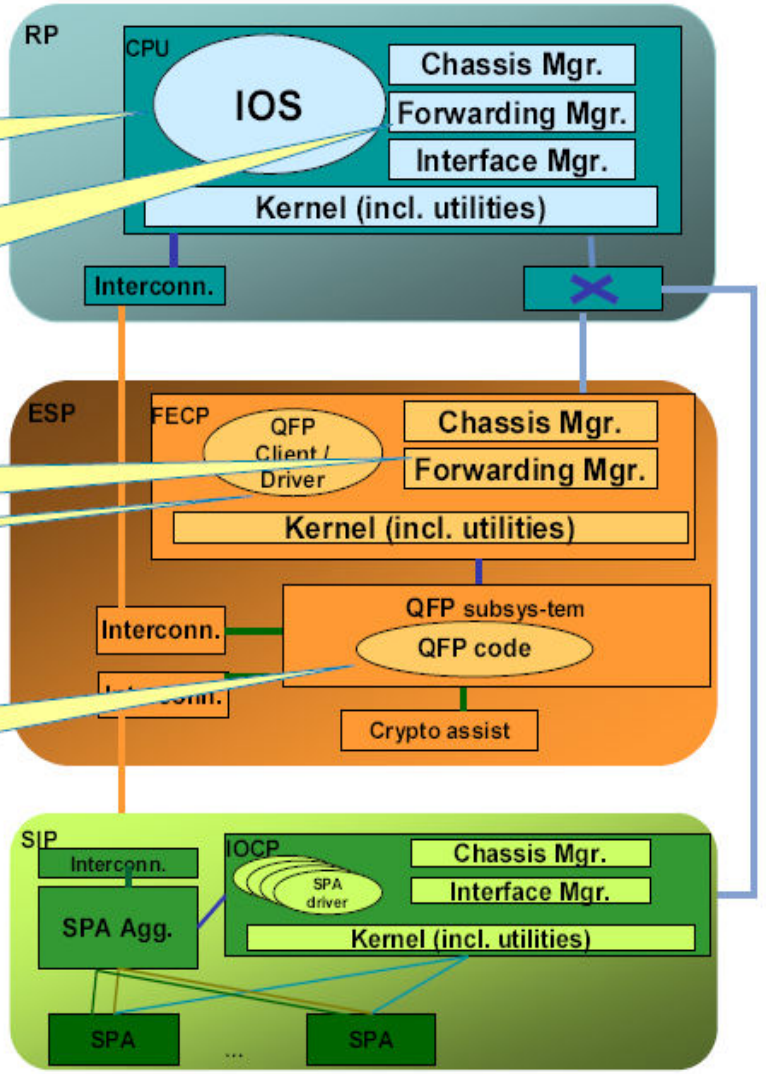
- Runs Control Plane
- Generates configurations
- Populates and maintains routing tables (RIB, FIB…)

- Provides abstraction layer between hardware and IOS (manages ESP redundancy)
- Maintains copy of FIB and interface list
- Communicates FIB status to active & standby ESP (or bulk-download state info in case of restart)

- Communicates with Forwarding manager on RP
- Provides interface to QFP Client / Driver

- Maintains copy of FIBs
- Programs QFP forwarding plane and QFP DRAM
- Statistics collection and communication to RP

- Implements forwarding plane
- Programs PPEs with forwarding information



Software Architecture – IOS XE

Kernel

- Control CPUs (RP, FECP, IOCP) run a Linux operating system Kernel

  Responsible for process scheduling, memory management, interrupts …

- Also includes a suite of low-level applications

  e.g. allow console access for debugging, SNTP, OBFL

  common for the base software, but may vary between the different control CPUs

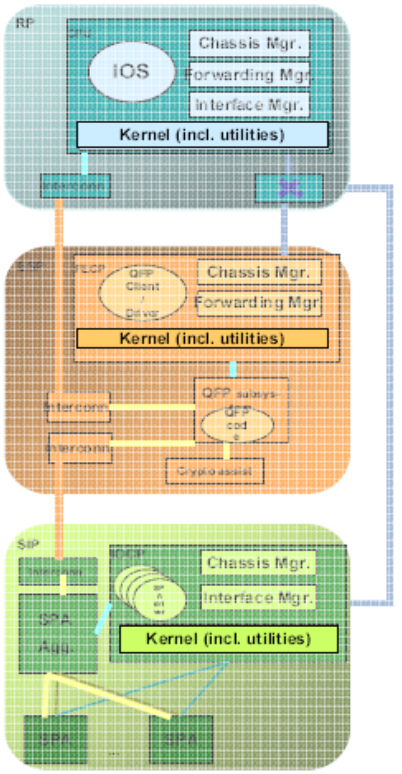
- Provides connectivity to other system components via IPC

  Code includes device drivers for EOBC or Hypertransport

  Kernel is responsible for directing IPC messages to the respective other software processes (IOS, chassis manager etc.)

  Implements punt-path for legacy data packets

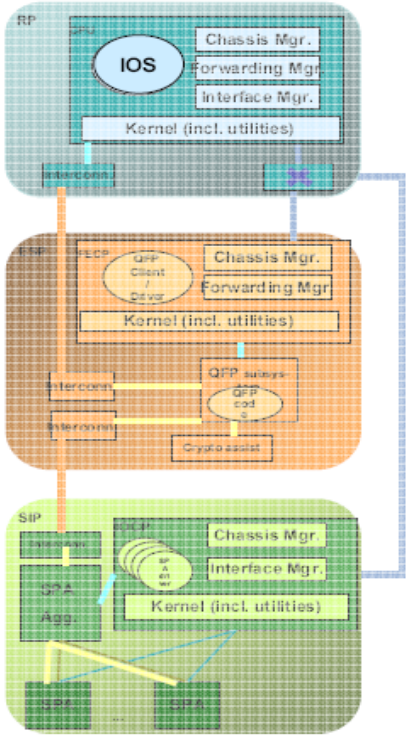
- Pre-emptible (can interrupt & prioritize processes)



Software Architecture – IOS XE

IOS

- Runs as a process under the Linux Kernel
  - IOS timing is governed by Linux Kernel scheduling
- Provides virtualized management ports
  - … since these are managed by their respective software processes
  - No direct hardware component access!
- Communicates with other software processes via IPC
- Runs Control plane features
  - CLI and configuration processing
  - SNMP handling
  - Running routing protocols & computing routes
  - Managing interfaces and tunnels
  - Session management
- Processing of punted features (legacy protocols)
- Two IOS processes can run in parallel for software redundancy on 2RU and 4RU systems
- Based on IOS 12.2SR features, which includes 12.2SB and some 12.4T-based features



Software Architecture – IOS XE

QFP Client / Driver and µcode

- QFP Client

  Allocates and manages resources on QFP (data structures, memory, scheduling hierarchy)

  Receives requests from IOS via RP

  Re-initializes QFP and its memory if a software error occurs

- QFP Driver

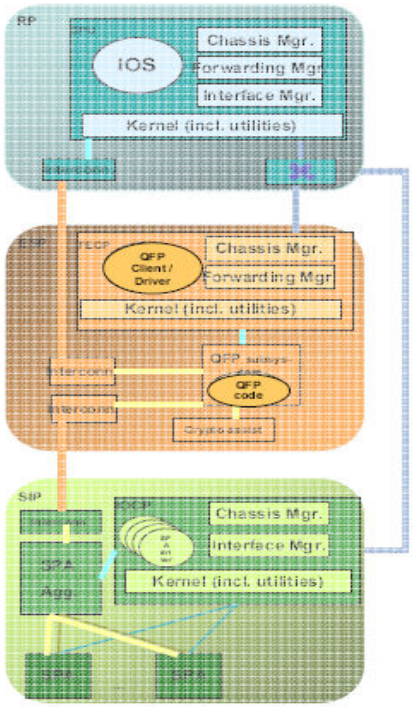
  Provides low-level access and control to QFP (register access)

  Provides communication path between QFP client and QFP via IPC

- QFP microcode (µcode)

  Implements data plane on PPEs

  Feature Invocation Array determines feature ordering



结束语

谢谢大家！

欢迎访问弯曲评论

www.tektalk.cn

彎曲評論

科技 · 人物 · 潮流