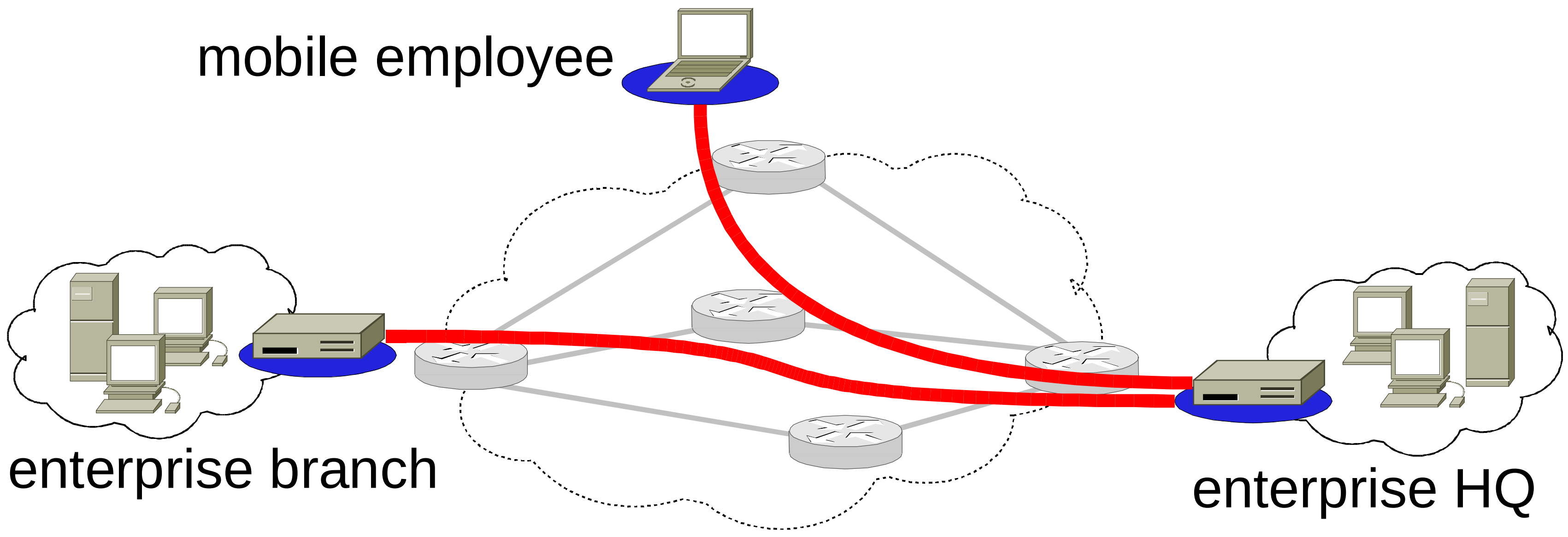
Advanced Virtual Private Network Support on FreeBSD systems

Riccardo Scandariato, Fulvio Risso
Politecnico di Torino, Italy

Outline

- PPVPN definition
- Needed support for PPVPN
- Roadmap of modifications
- Implementation details (FreeBSD 4.4)
- Conclusions

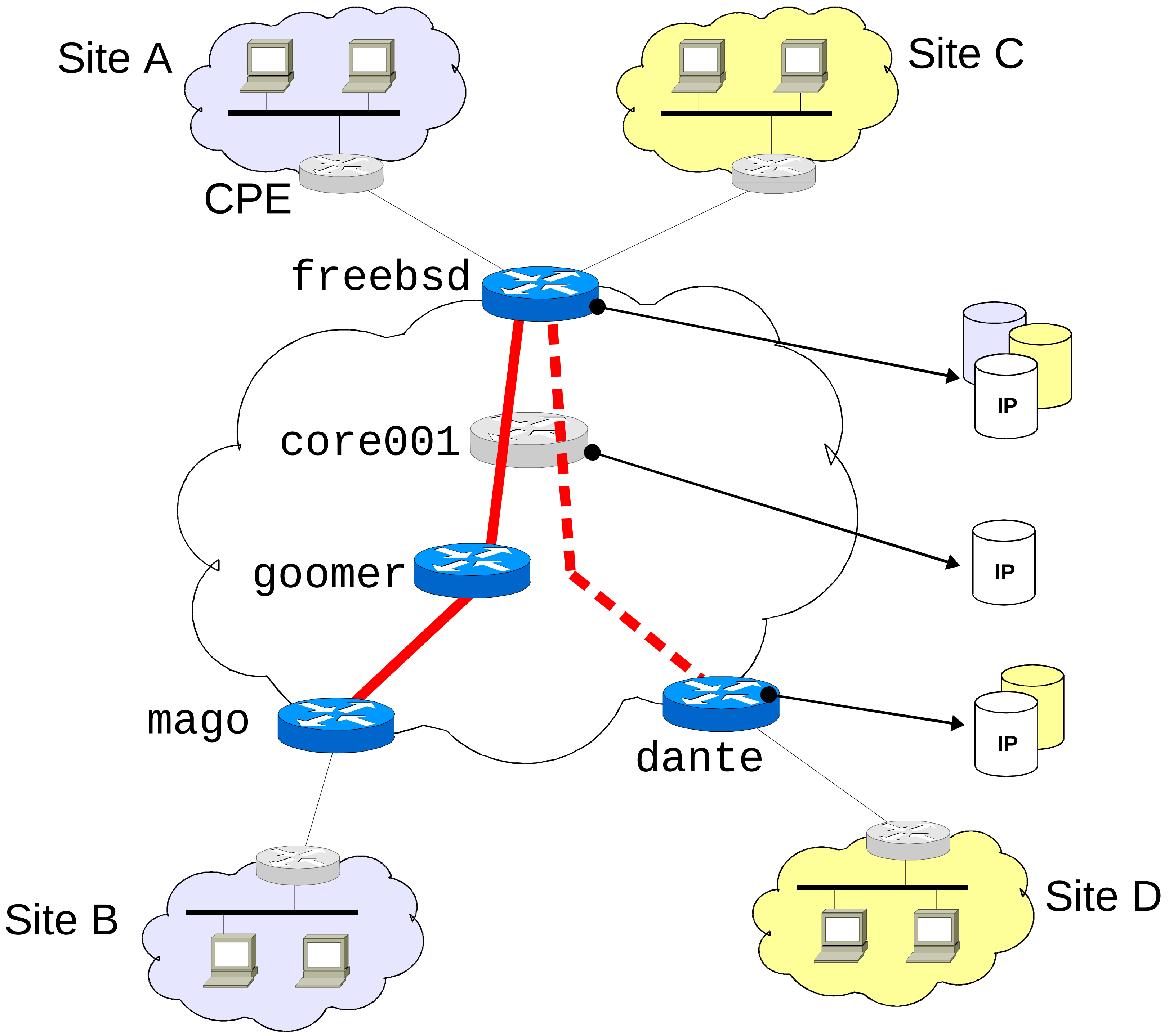
Customer-based VPN



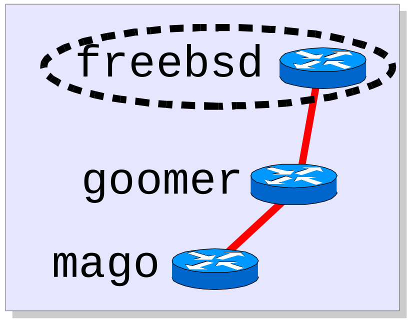
- VPN connectivity supported by customer equipment
- Network provider just as transport (VPN-unaware)

Provider Provisioned VPN

- VPN connectivity supported by the provider network
  - Transparency to the end-user
- Multiple virtual network concurrently deployed on the same physical network
  - Routers shared among different VPNs
- Addresses are chosen by clients (typically out from the private space)
  - Overlaps and collisions across VPNs


Site A
Site C
CPE
freebsd
core001
goomer
mago
dante
IP
IP
IP
Site B
Site D

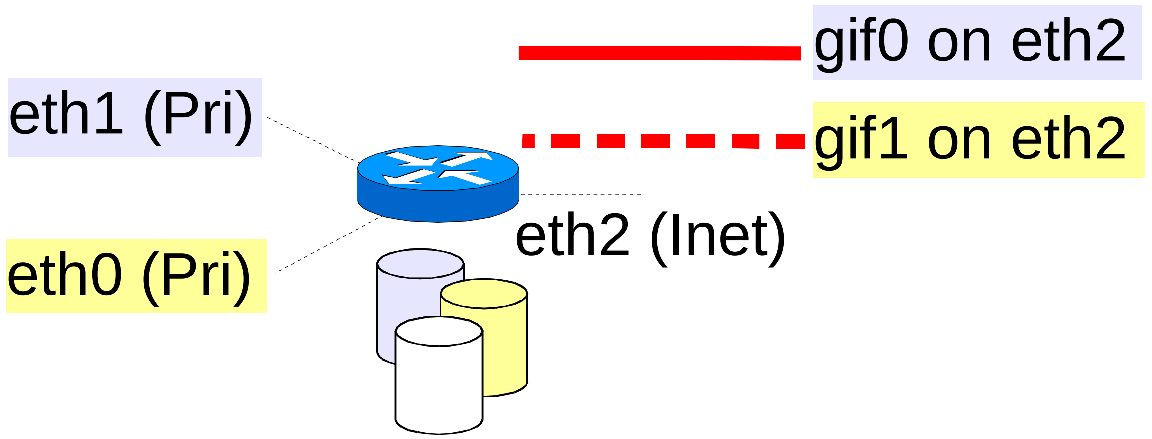
Access VPN router



Identification

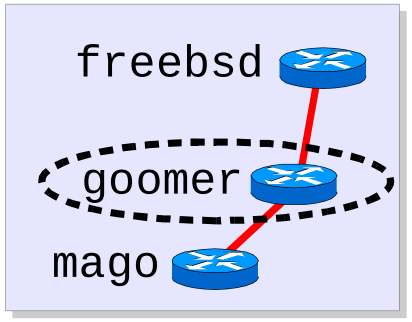
Encapsulation

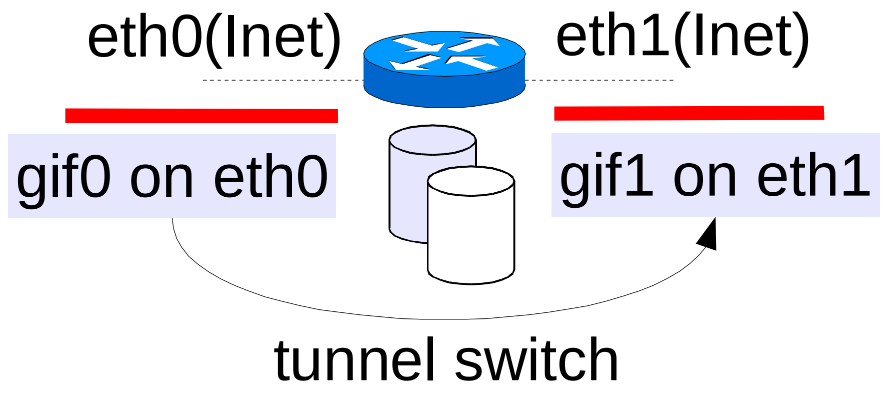
Vx lookup



Inet SRC	Inet DST	VPN SRC	VPN DST	
freebsd(eth2)	goomer(eth0)	10.0.1.1	10.0.2.7	payload

Core VPN router





Inet SRC	Inet DST	VPN SRC	VPN DST	
freebsd(eth2)	goomer(eth0)	10.0.1.1	10.0.2.7	payload

⬇

Inet SRC	Inet DST	VPN SRC	VPN DST	
goomer(eth1)	mago	10.0.1.1	10.0.2.7	payload

Tunneling

- IP-in-IP already provided by FreeBSD (`gif` pseudo-interfaces)
- Paired Point-to-Point numbered links

  -
    ```
    freebsd# ifcongig gif0 create
    ```

  -
    ```
    freebsd# ifconfig gif0
      inet 10.0.0.1 10.0.0.2
      netmask 255.255.255.0
    ```

  -
    ```
    freebsd# gifconfig gif0
      inet 130.192.31.1 130.192.31.2
    ```
  - Same on peer

Summing up

- Many nets with their own topologies
- Same routers serving many nets
- No assumption about address spaces
  - Cope with overlapped address spaces
- Each packet must be forwarded according to the pertaining VPN

Rationale

- Routing table virtualization
  - *Introduced by this work*
  - Forwarding virtualization
  - Routing virtualization
- Tunneling (IP-in-IP)
  - Already provided by FreeBSD (see issues...)
- *Commitment*
  - *As few modifications as possible*
  - *Harmonize with existing code*
  - *The simpler the better!*

Modified files

```
sys/sys/socket.h
sys/sys/socketvar.h
sys/sys/sockio.h
sys/kern/uipc_socket.c
sys/kern/sys_socket.c
sys/net/if_var.h
sys/net/if.h
sys/net/if.c
sys/net/route.h
sys/net/route.c
sys/net/raw_cb.h
sys/net/rtsock.c
sys/net/raw_usrreq.c
sys/netinet/ip_input.c
sys/netinet/if_ether.c
```

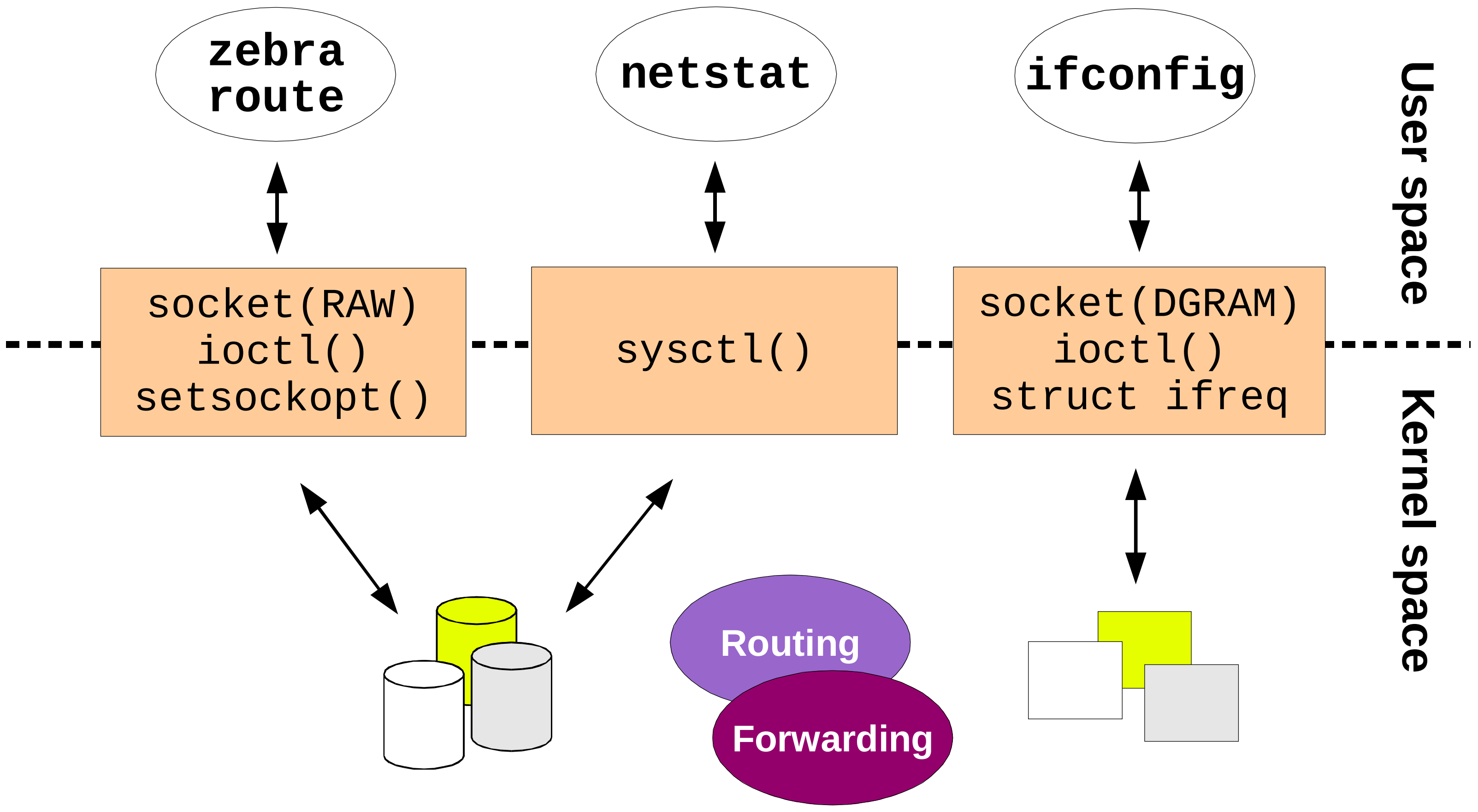
```
netstat/netstat.h
netstat/route.c
netstat/main.c
```

```
route/keywords
route/route.c
```

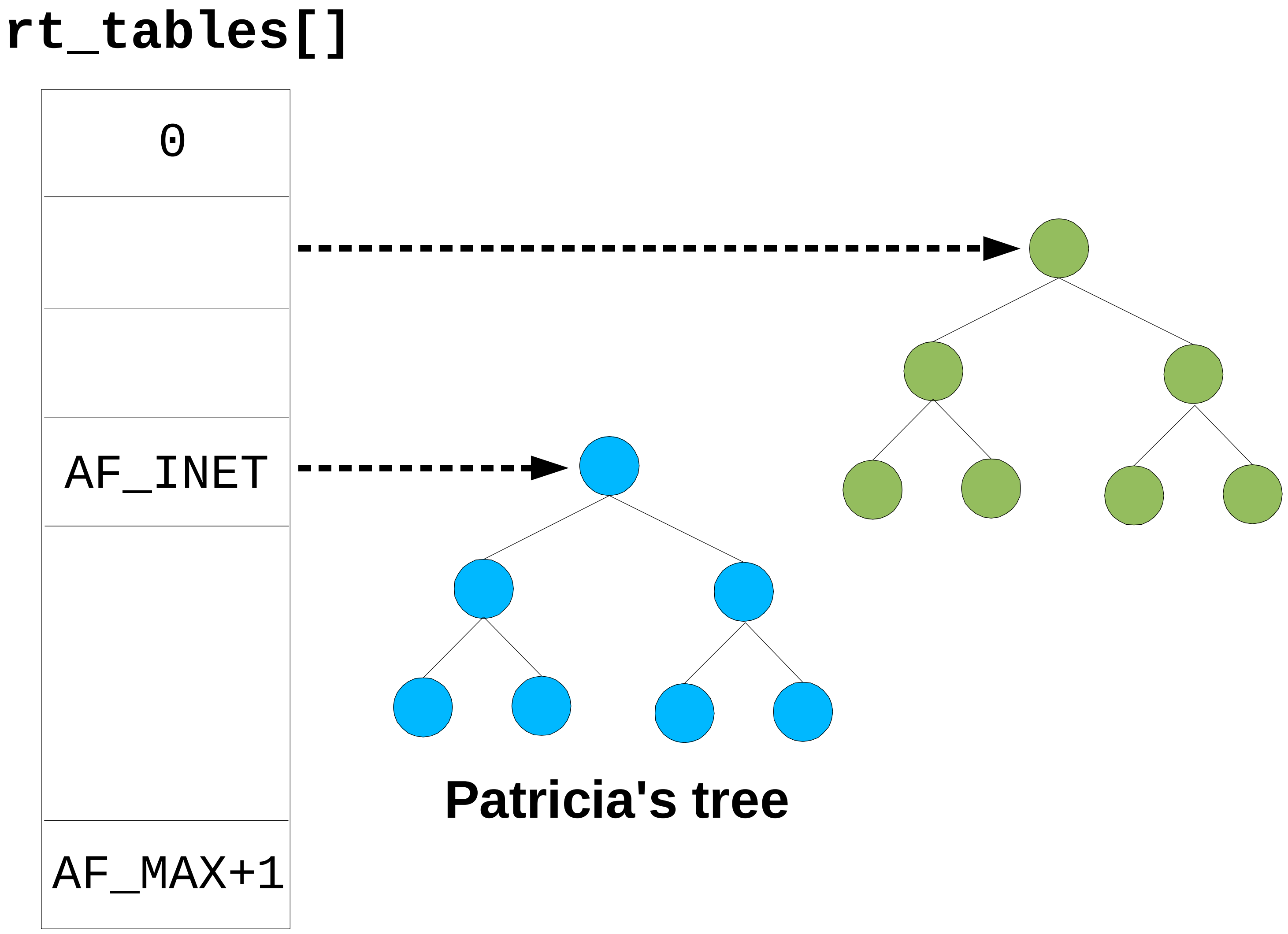
```
ifconfig/ifconfig.c
```

```
zebra/lib/vpn.h
zebra/main.c
zebra/kernel_socket.c
zebra/rtread_sysctl.c
```

Roadmap


zebra
route
netstat
ifconfig
User space
socket(RAW)
ioctl()
setsockopt()
sysctl()
socket(DGRAM)
ioctl()
struct ifreq
Kernel space
Routing
Forwarding

Multiple routing tables

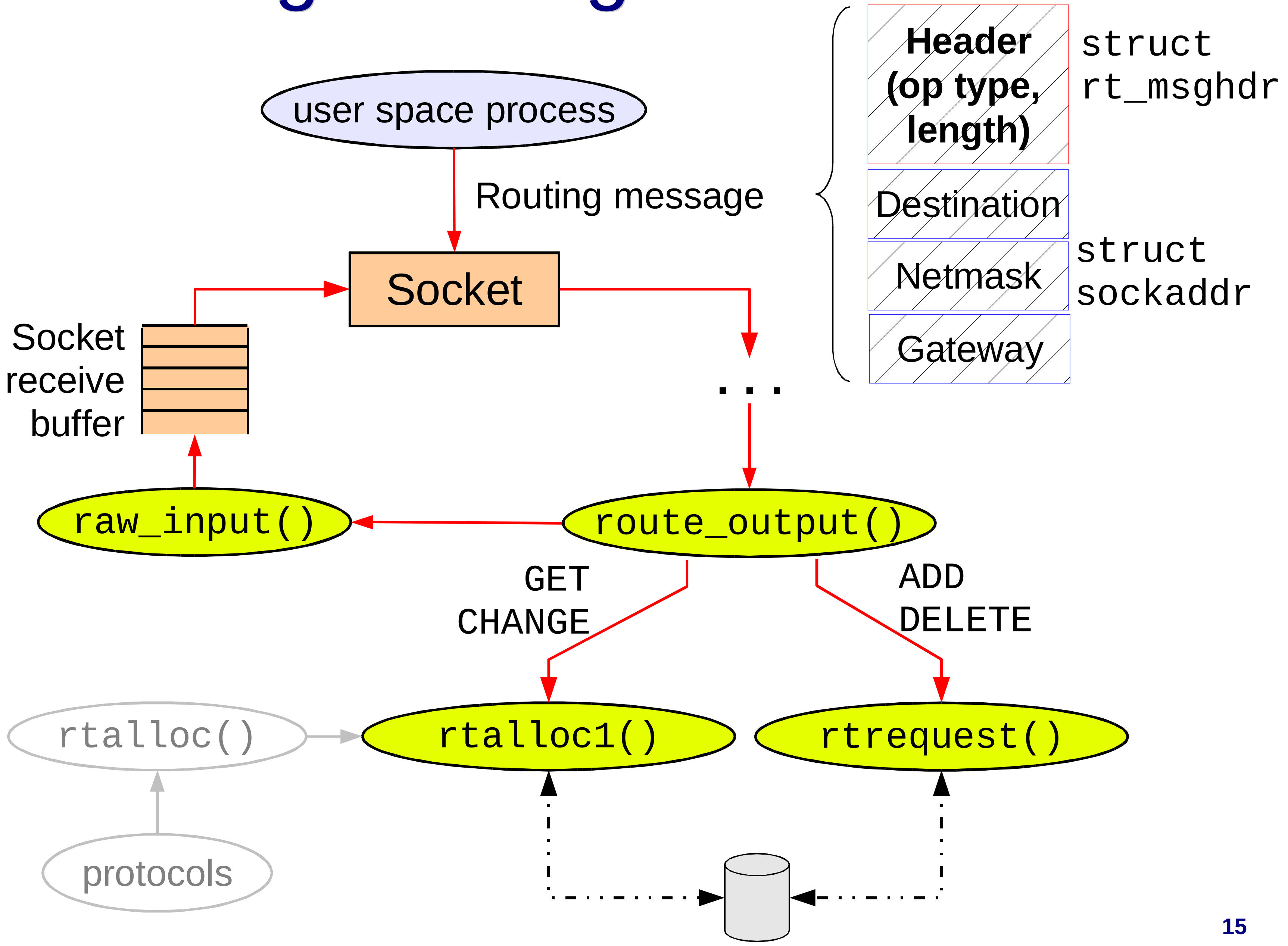


Patricia's tree

Multiple routing tables cont'd

- **`vpn_rt_tables[VPN_MAX + 1]`**
  - `VPN_MAX` defined in sys/socket.h
- Array statically allocated (net/route.c) for efficiency
- Tables dynamically initialized on demand the first time they are accessed
  - `route_output(RTM_ADD) =>`
  - `vpn_rtrequest(RTM_ADD,vpnid) =>`
  - `rn_inithead(&vpn_rt_tables[vpnid])`

Routing messages



Routing sockets

- VPN ID added to socket structure (sys/socketvar.h)
  - **`struct socket{ u_int vpnid; }`**
- VPN ID field initialized to zero when socket is created by `socket()` sys call
  - **`socreate()`** (kern/uipc_socket.c)

Routing sockets cont'd

- VPN ID can be set through the `SO_VPNID` option (sys/socket.h) of `setsockopt()`
  - **`sosetopt()`**, **`sogetopt()`** (kern/uipc_socket.c)

- VPN ID can be also set through the `SIOC(G,S)VPNID` options (sys/sockio.h) of `ioctl()`
  - **`soo_ioctl()`** (kern/sys_socket.c)

Table interaction

- **`route_output()`** (net/rtsock.c)
  - `RTM_ADD` and `RTM_DELETE` now call **`vpn_rtrequest()`** (net/route.h,c)
  - `RTM_GET` now selects the table based on the socket's vpnid before `rnh_lookup()`

Routing messages from kernel

- VPN ID added as argument to `raw_input()`
  - **`vpn_raw_input()`** (`net/raw_cb.h`, `net/raw_usrreq.c`)
- Message is now delivered only to routing sockets with the same VPN ID

Sysctl

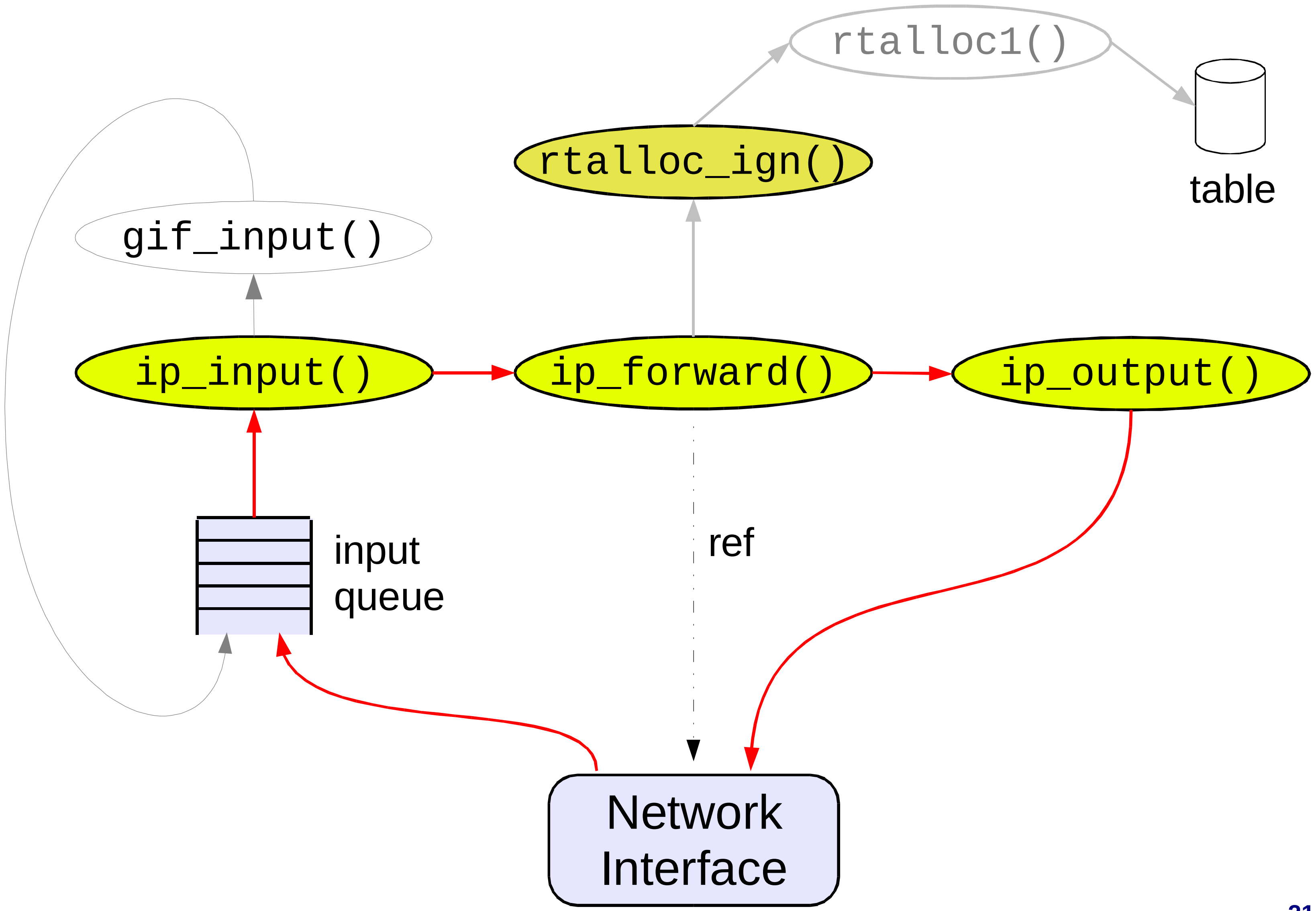
- E.g. used by `netstat` to read the whole table
  - **`sysctl_rtsock()`** (net/rtsock.c)

- Example

  -
    ```
    struct rt_msghdr *msg;
    int mib[6] = {CTL_NET, PF_ROUTE,
                  0, AF_INET,
                  NET_RT_DUMP, 7}
    sysctl(mib,msg);
    ```

VPN ID (added)

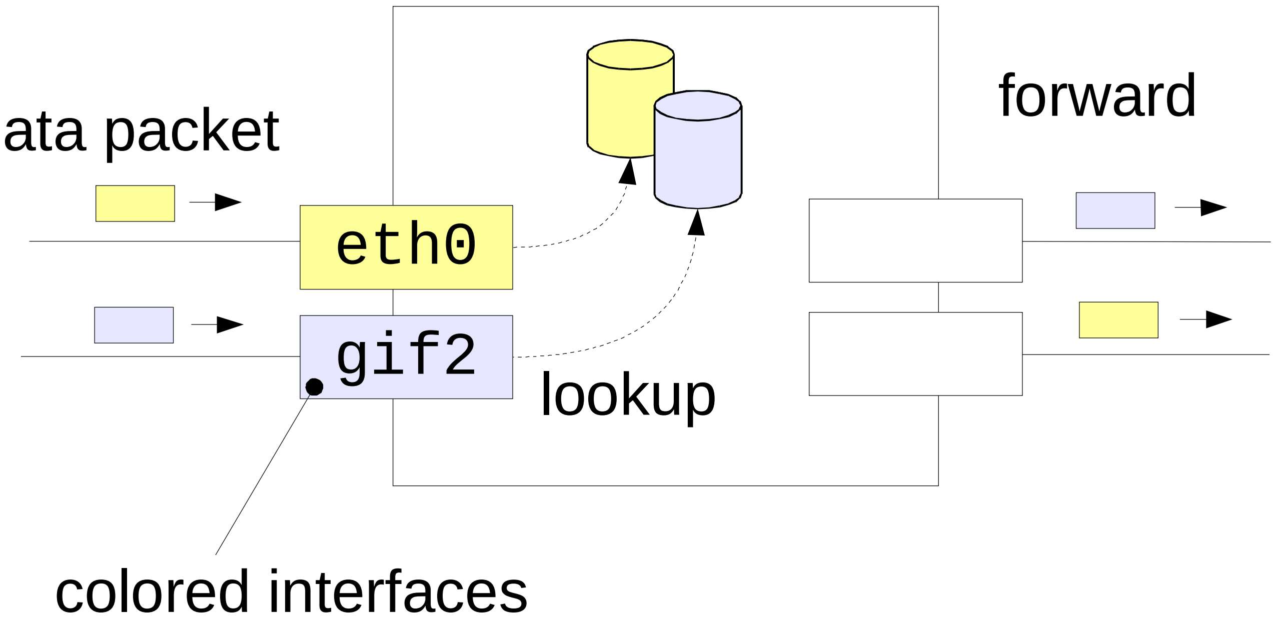
Packet forwarding process



Forwarding virtualization

- **`ip_forward()`** (netinet/ip_input.c)
  - VPN ID is retrieved from the receiving interface (either physical or pseudo)
  - It now calls **`vpn_rtalloc_ign()`** (net/route.h,c)
- Ancillary functions
  - **`vpn_rtalloc()`**, **`vpn_rtalloc1()`** (net/route.h,c)

Traffic identification



Interface marking

- VPN ID added to interface structure (net/if_var.h)
  - **`struct ifnet{ u_int if_vpnid; }`**
- VPN ID field initialized to zero when interfaces are created at boot
  - **`if_attach()`** (net/if.c)

Interface marking cont'd

- VPN ID can be set through the `SIOC(S,G)IFVPNID` options (sys/sockio.h) of `ioctl()`
  - **`struct ifreq{ u_int ifr_vpnid; }`** (net/if.h)
  - **`ifioctl()`** (net/if.c)

User space programs

- `route add default freebsd.polito.it -vpn 7`
- `netstat -v 7`
- `ifconfig gif0 10.0.0.1 netmask 255.255.255.0 vpnid 7`
- `zebra -f zebra.mago.7.conf -V 7`
  - `ospfd -f ospfd.mago.7.conf`

Issues (i)

- ARP cache update not virtualized
  - ARP lookup is virtualized (netinet/if_ether.c)
  - ARP entries still written into base table
  - Issue does not affect if a L3 CPE is used between the destination and the egress router

Issues (ii)

- `gif` interfaces are colored to identify the pertaining VPN
- Different VPNs between the same couple of nodes need different tunnels/`gifs`
- Incoming `gif` is recognized through outer src address and outer dst address
    - No multiple IP-in-IP tunnels between the same couple of physical interfaces (addresses)
- GRE (with KEY field) can be used to disambiguate

Improvements

- VPN identification at ingress points
  - Fine grained traffic filters
  - Colors are better for `gif` interfaces
- Zebra support
  - VPN_ID in communication protocol between `ospfd` daemons and the `zebra` router manager
- Secure transport of VPN traffic: IPSec
- Per-VPN QoS warranties: ALTQ

Info

- Do you wanna try it?
  - http://softeng.polito.it/freebsd/
- Do you wanna know more details?
  - **Riccardo Scandariato,** scandariato@polito.it
  - **Fulvio Risso,** risso@polito.it

Q&A