Hybrid Routing

Combining the power of hardware and flexibility of software routers

Koen van Hove
University of Twente

Hardware

- Specialised hardware
- Fast
- Inflexible

Software

- Commodity hardware
- Slower
- Flexible

Prerequisite for hybridisation: open hardware and software

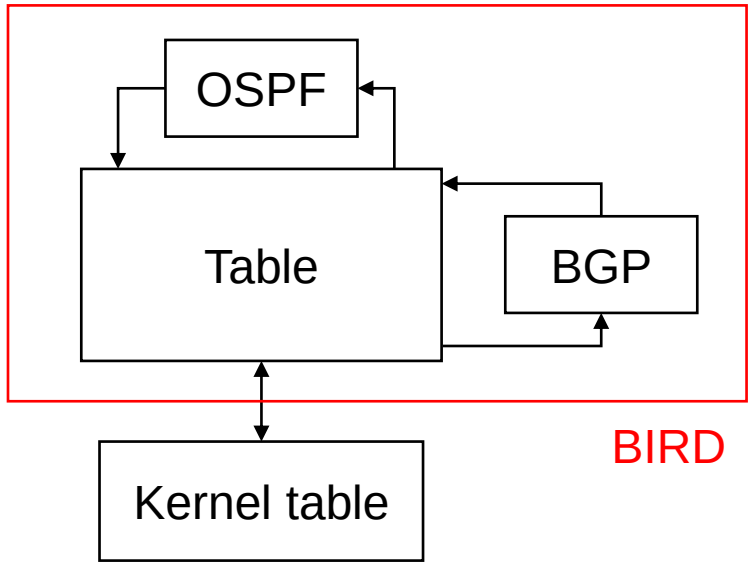
Open hardware and software

- P4-capable switches
- BIRD and Free Range Routing

P4

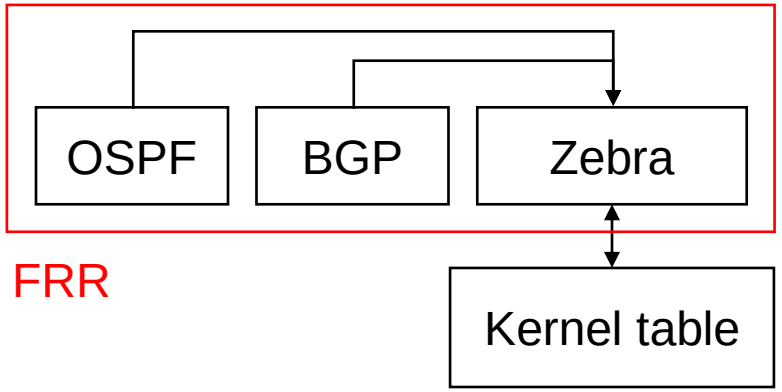
- “A language for programming the data plane of network devices”
  - A descriptive language for hardware
- Lacking flexibility especially w.r.t. cryptography (unless already implemented in the hardware)

BIRD



- Single daemon
- All protocols integrated
- Keeps its own table that synchronises with the kernel routing table

Free Range Routing



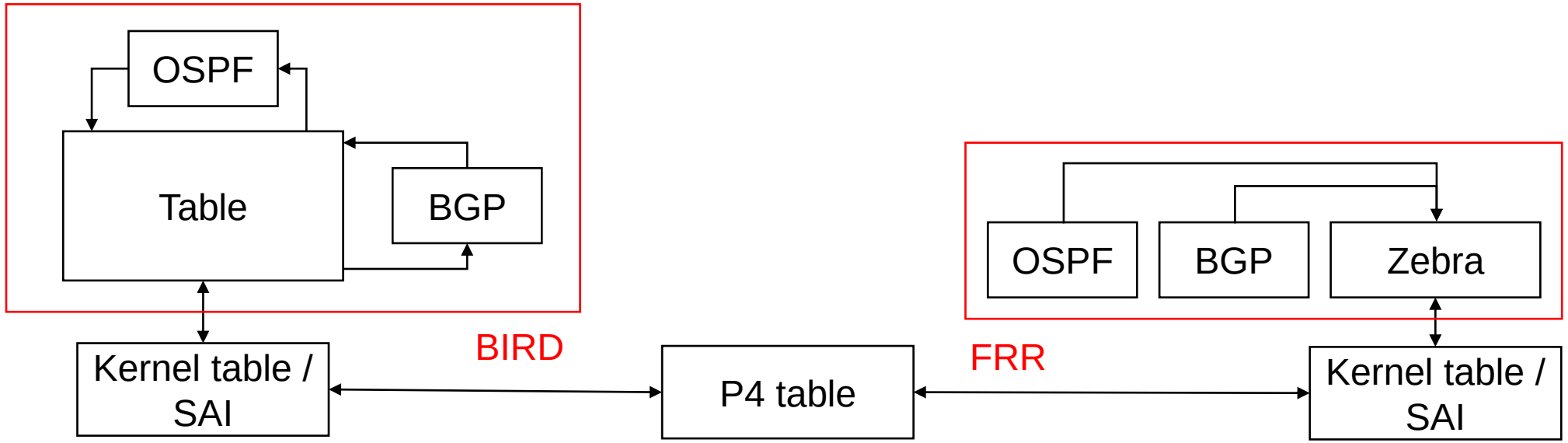
- One daemon per protocol
- Protocols interact independently from each other
- Zebra is also a daemon that translates routing information to the kernel table

Hybridisation

- A question of flexibility where
- Different options provide and limit flexibility in different areas

Kernel synchronisation

- Synchronise the kernel table with the P4 table



- BIRD, FRR, or any other software router will work
- Testing with new routing protocols requires integrating it in either BIRD or FRR

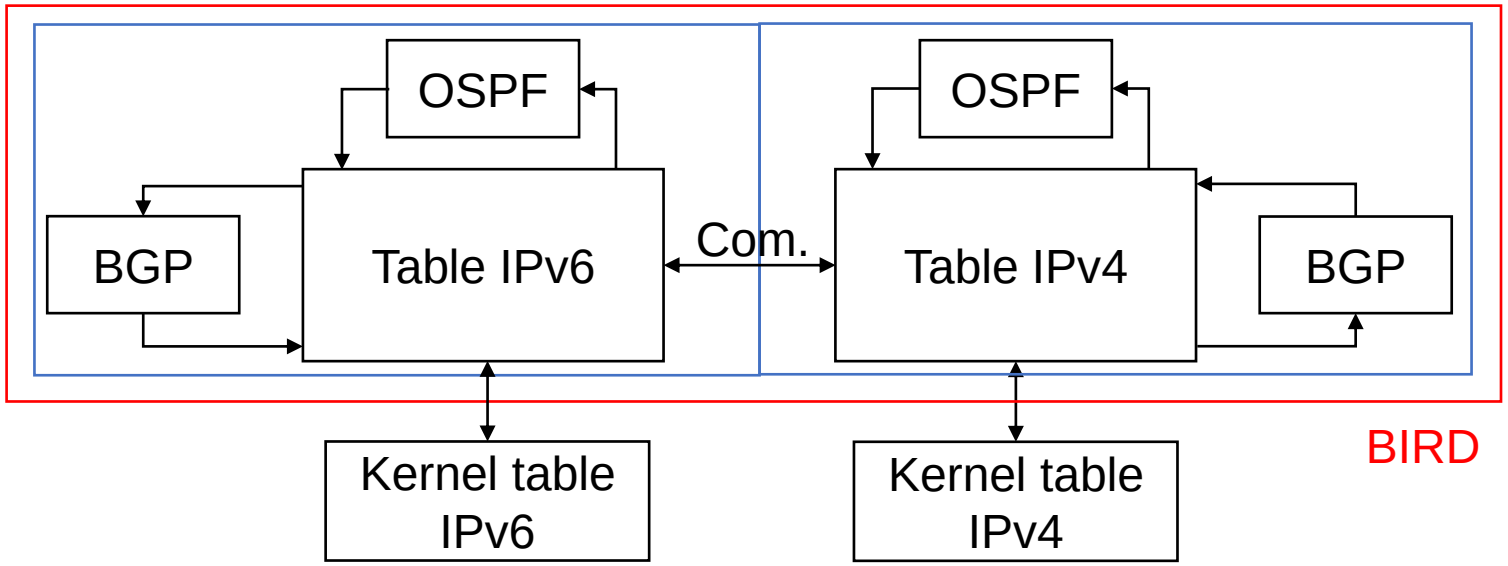
Kernel synchronisation

- Works well with new routing protocols that do not modify IPv4/IPv6
- However, this would also work with any router that allows the routing table to be populated via e.g. serial

- But what if we do not run standard IP?

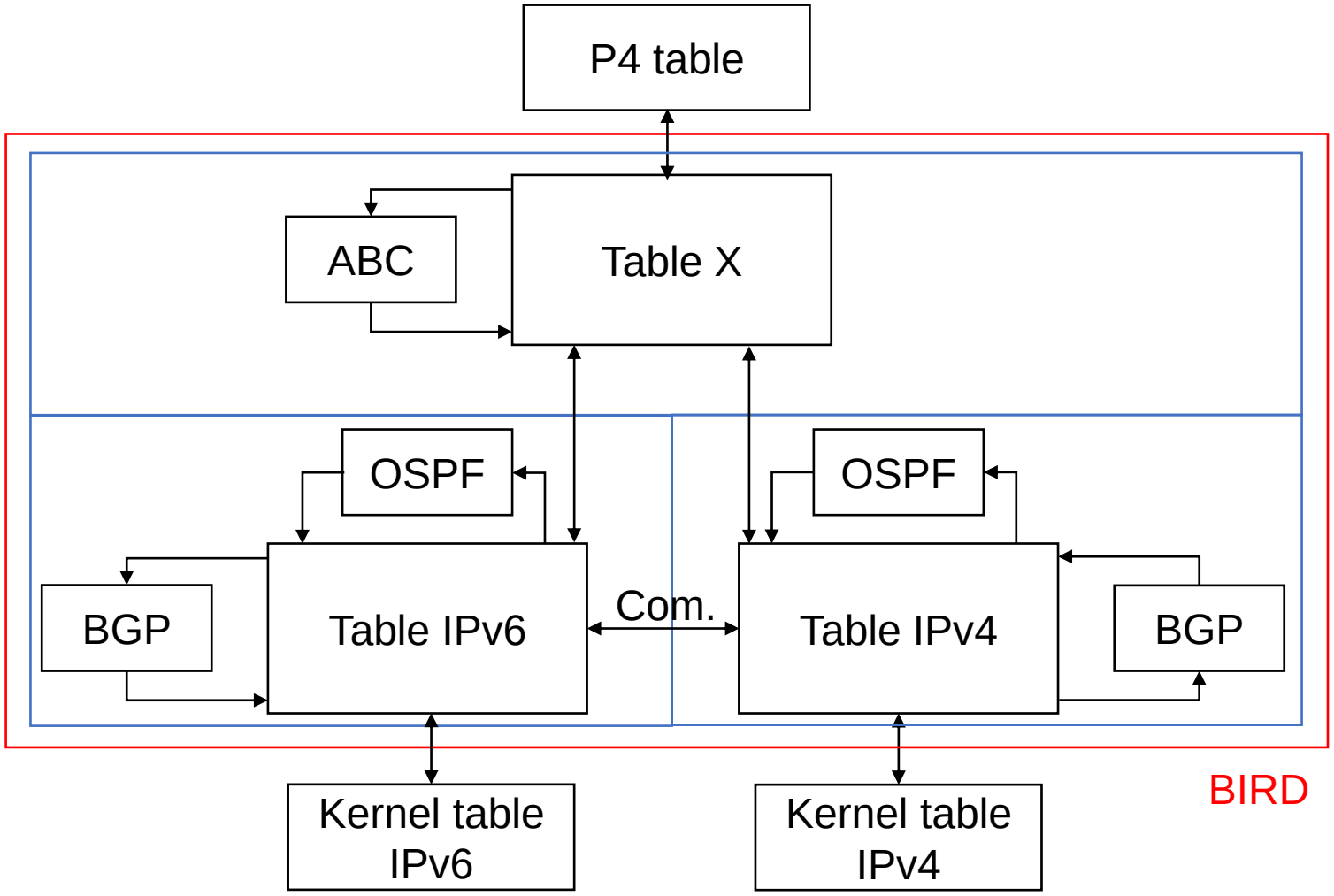
Dependence on IP: BIRD

- Separate table and daemon for IPv4/IPv6
- Separate config too
- Communication channel between them



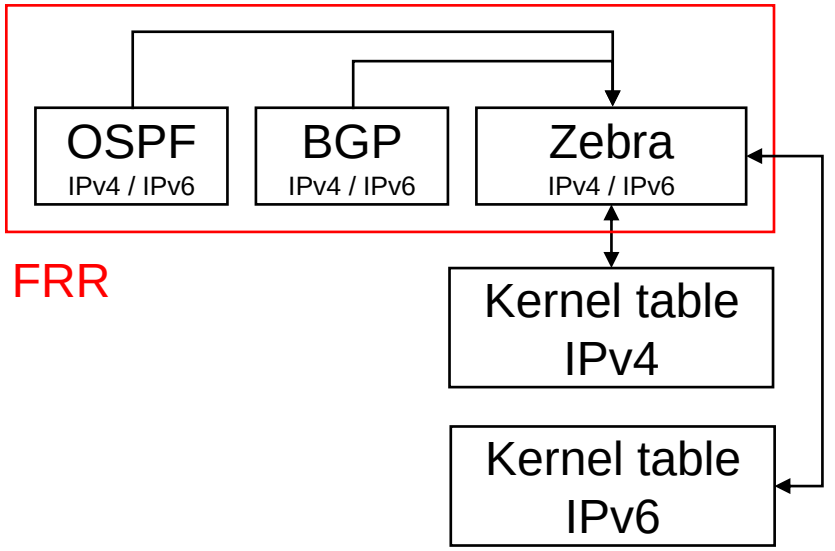
Dependence on IP: BIRD

- Room for a third one is a similar vein (dubbed X)
- That one can be synced directly with P4, as there is no need to create an X BIRD table and X kernel table



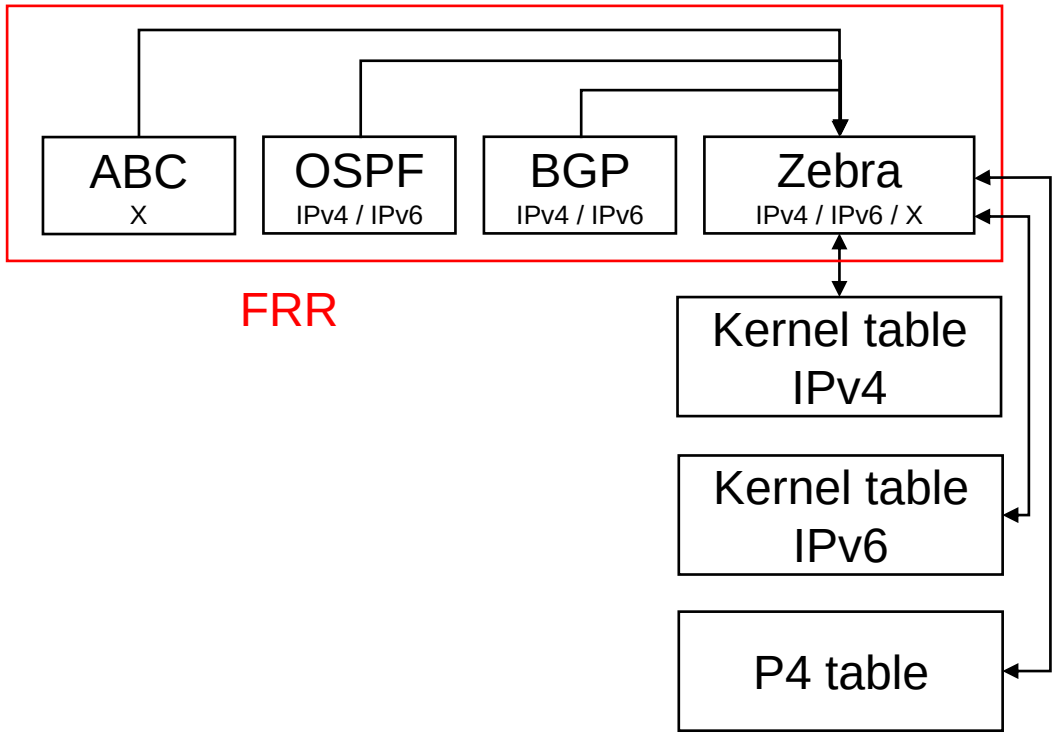
Dependence on IP: FRR

- Same daemon for IPv4 and IPv6
- Closely tied in with the kernel table
- No separate table management by Zebra, it directly interacts with the Kernel table



Dependence on IP: FRR

- Create a new daemon for our ABC routing protocol
- Adapt Zebra for X
- Adapt Zebra for the P4 table



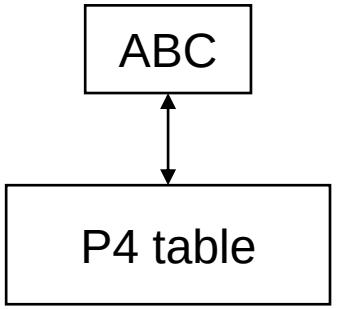
Dependence on IP

- Works well in conjunction with existing routing protocols
- Also works well for adaptations of existing protocols

- May be excessive for greenfield protocols

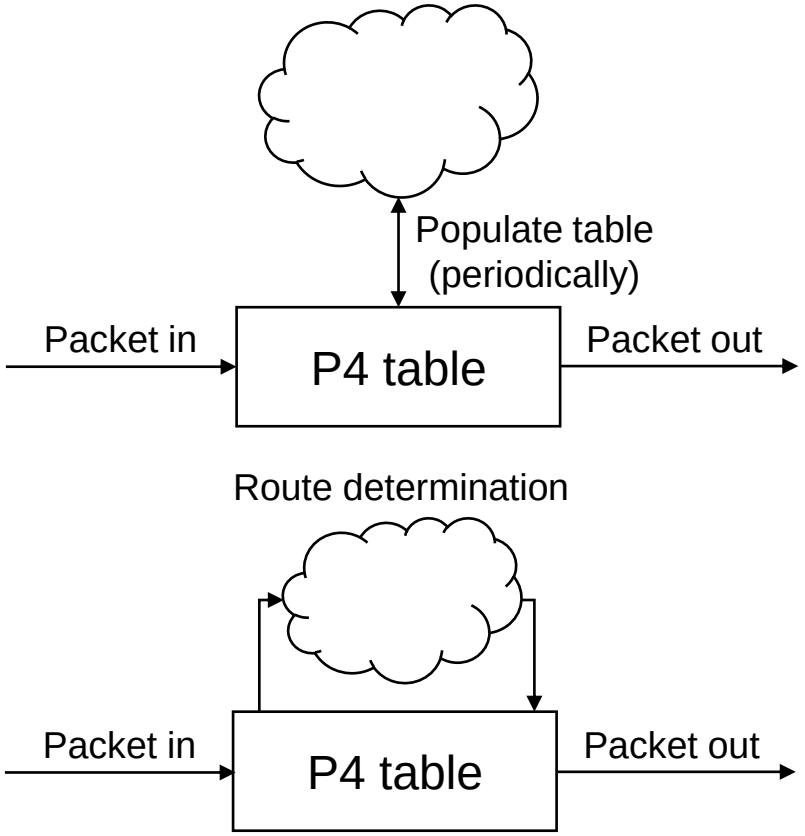
Greenfield protocols

- No need to adapt existing software for new protocols
- Assumes no protocol manager is necessary
- Very simple, and very flexible
- About as useful as a blank canvas
- May cause issues in the long run



Adaptive forwarding

- So far we have assumed that we have routes that are updated periodically
- What if we want to do routing adaptively in just-in-time fashion (e.g. SCION)?
- BIRD's and FRR's design do not support this
- Greenfield may prove useful in this case

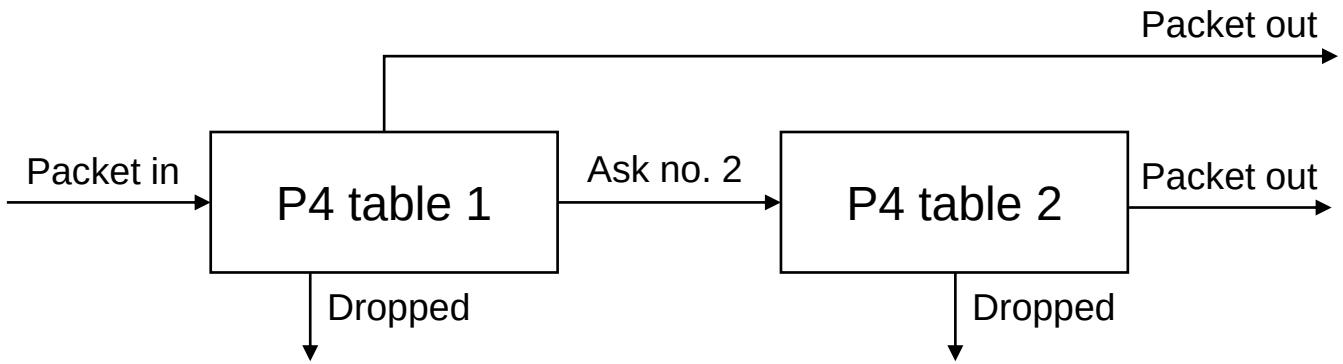


Summary

- Combining the power of hardware and flexibility of software routers
- Different hybridisation options provide and limit flexibility in different areas:
  - Using the kernel forwarding table/switch abstraction interface
  - Directly communicating from BIRD/FRR to P4
  - Bypass BIRD/FRR altogether

Multiple P4 routers in line

- What if my forwarding table size is not sufficient



Hand-off to software routing

- What if my forwarding table size is nowhere close to sufficient