

The Flock Routing Development Crate

24th September 2022

Flock Networks Ltd
160 Kemp House,
London, EC1V 2NX

info@flocknetworks.com

Product Overview

The Flock "Routing Development Crate" (RDC) allows customers to quickly create a secure, performant and highly customizable IP routing suite that is easy to embed in any network device.

The RDC source code is available to license which enables network device manufacturers and network operators to rapidly create and deploy customized internet devices such as IP routers, IP gateways and "Internet of Things" (IOT) devices. Internet devices developed using RDC have been deployed since 2021, and have proved to be reliable and efficient.

The RDC source code is a platform agnostic library which implements a comprehensive set of internet RFC standards, a generic configuration and operation API, and a generic route programming API. Also included is a reference application that uses the RDC library to implement a fully featured IP routing suite that runs on the Linux or SONiC platforms.

The RDC source code is designed to be highly flexible. RDC enables easy compile time customization and extension of the included protocol implementations, by providing embedded "modify hook" points in the source code. RDC is compiled to match the hardware capabilities of the product that is being developed, from the lowest end consumer devices to the highest end internet route servers and Path Computation Element (PCE) servers.

RDC is designed to scale at run time to match the available CPU cores. On a router with four logical CPU cores it can signal one billion BGP route updates across one thousand BGP neighbors in under 90s from startup. If that is too slow for your network requirements then just choose a router with more CPU cores.

Current IP routing suites were implemented decades ago. They no longer have a clean architecture so new feature development is very slow. They are written in unsafe languages such as C or C++, meaning a single incorrect line of code can crash the entire network device, or worse leave the network device running in an unknown and possibly insecure state.

RDC is written exclusively in the Rust programming language. We are not alone in believing all performance critical internet infrastructure will be migrated to Rust.

> *"Speaking of languages, it's time to halt starting any new projects in C/C++ and use Rust for those scenarios where a non-GC language is required. For the sake of security and reliability. the industry should declare those languages as deprecated."*
> Mark Russinovich, CTO Microsoft Azure (19th September 2022)

Routing Development Crate Contents

RDC contains the complete source code for the components shown in blue below. Customers are able to customize the implementation by implementing the green components. Some of the advantages of using RDC are numbered, with explanations below.

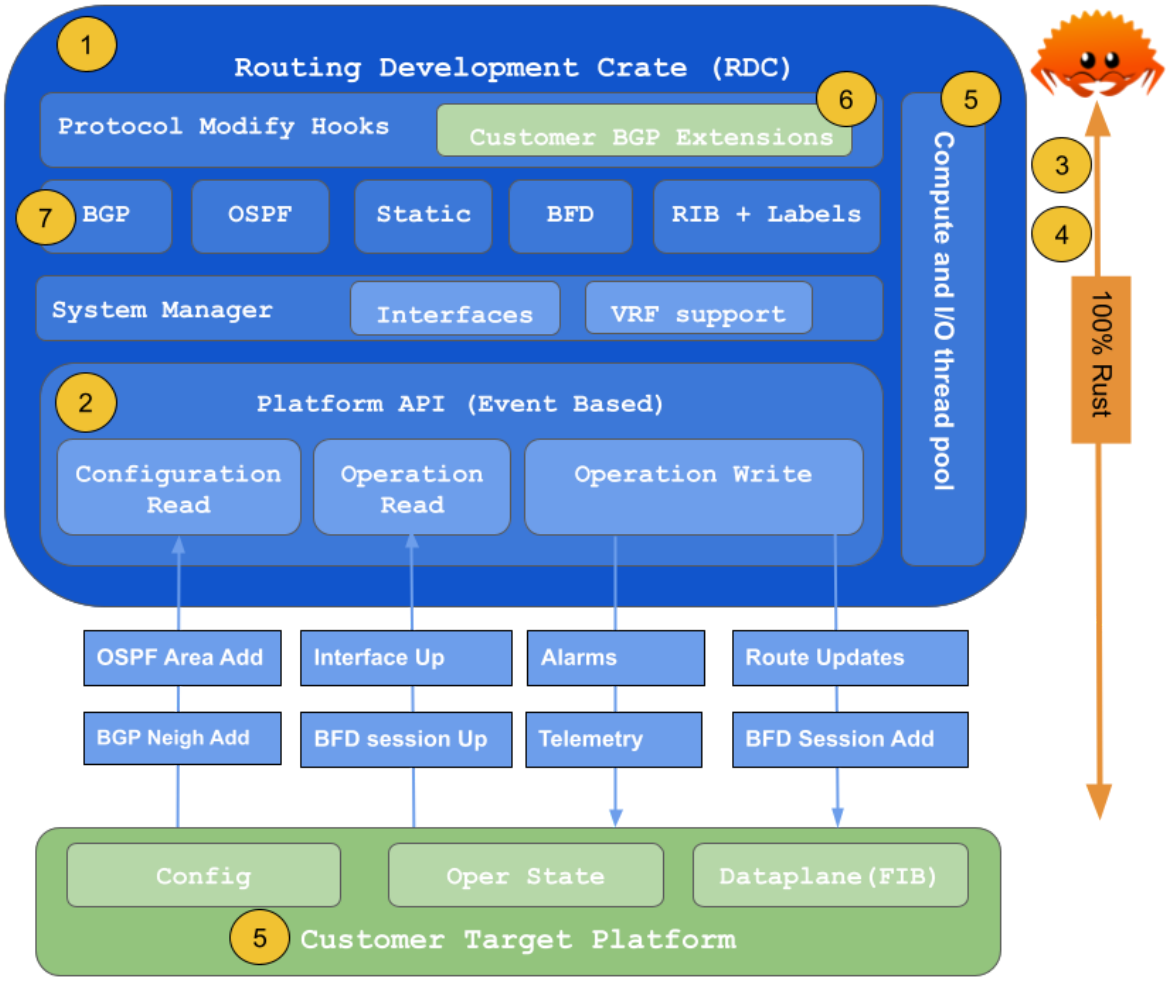


Figure 1: *RDC Contents*

RDC Advantages

1. **Clean architecture**: RDC has been designed and developed from scratch by a team of highly experienced IP protocol experts, developers and testers. The best practices from the past 30 years of IP router design have been used, and designs that have historically proved to be unreliable and inflexible have been avoided.
2. **Easy customization:** RDC is highly modular and presents strong flexible API's, making it trivial to customize. The generic configuration and operation event based

API's allow RDC to be controlled and monitored by any external model (YANG, NETCONF, REST, JSON, CLI, Prometheus…). The generic route programming event based API can easily be customized to program any kernel / data plane. The provided Linux reference implementation supports programming via Netlink and FPM (SONiC).

3. **Highly secure**: RDC is entirely implemented in the Rust language. Rust is the only available memory safe systems programming language. Implementing applications in Rust has been shown to reduce security vulnerabilities by 70%. This security guarantee is provided at compile time so these bugs cannot exist in the deployed device. RDC is the first and currently only IP routing suite written in Rust.
4. **Highly performant**: The Rust language compile time memory safety guarantees means RDC can run in a single operating system process. This means all the code has safe access to the same shared heap memory. RDC does not need to perform any inter-process communication (IPC) making it far faster than other implementations. Rust also guarantees the absence of inter-thread "data races" at compile time. This eliminates the hardest problems when writing multithreaded code. RDC has been designed to scale linearly vs the number of CPU cores available. RDC is unique in that it can reliably utilize 64 cores in parallel whilst the network is converging, and then return to an idle state once converged. The RDC implementation is lock free which means each thread can progress without being blocked by any other threads. This also means RDC has a compile time guarantee that it cannot experience "dead lock", which can cause other implementations to fail at any time when running in production.
5. **Flexible to match the target platform**: RDC can be compiled to execute as:
    a. A single operating system thread, for use in low end devices
    b. A thread per protocol, for use in mid range devices
    c. A thread per protocol and a thread pool, for use in high end devices
6. **Extensible protocol behavior**: The RDC API also contains "modify hooks" embedded at key points in each protocol's implementation. The standard IETF protocol behavior can be modified or extended at compile time. For example, to implement a customized BGP "best path decision process", a single Rust function can be written to override the default best path function. Because this is done at compile time there is no run time performance cost to achieve the new behavior.
7. **Extensive Protocol Support**: RDC supports all the commonly used IETF RFC's for BGP, OSPFv2/v3, MPLS VPN, MPLS SR and BFD. The combination of the clean architecture and the expressiveness of the Rust language makes new feature development on RDC incredibly fast. RFC's that would take months to implement on legacy IP routing suites, can be implemented in a couple of weeks.

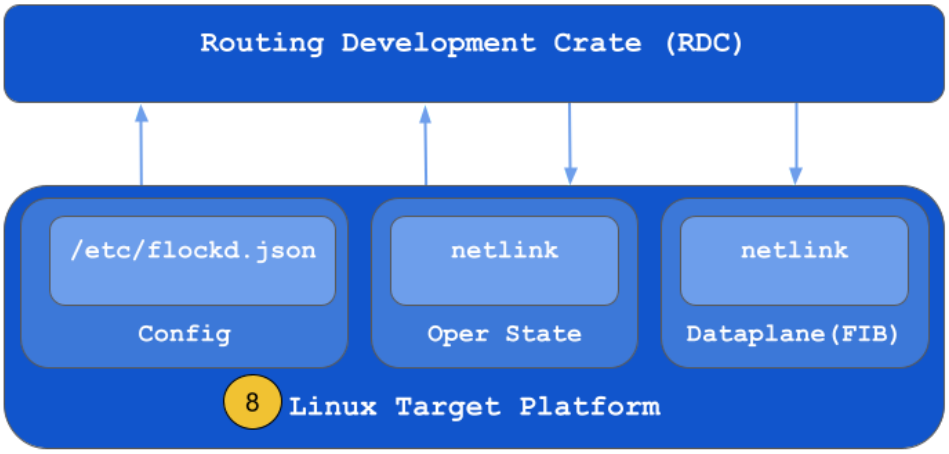


Figure 2: *RDC Reference Linux implementation*

8. **Complete:** RDC includes a production ready reference Linux implementation that works "Out of the box". If your product requires standard Linux IP Routing Suite behavior, similar to Free Range Routing (FRR) or BIRD, then you don't need to write any customization code. Just compile the reference implementation and deploy it. Alternatively this same implementation can be run on the SONiC network operating system, allowing a hardware forwarding / ASIC enabled network device to be quickly developed.

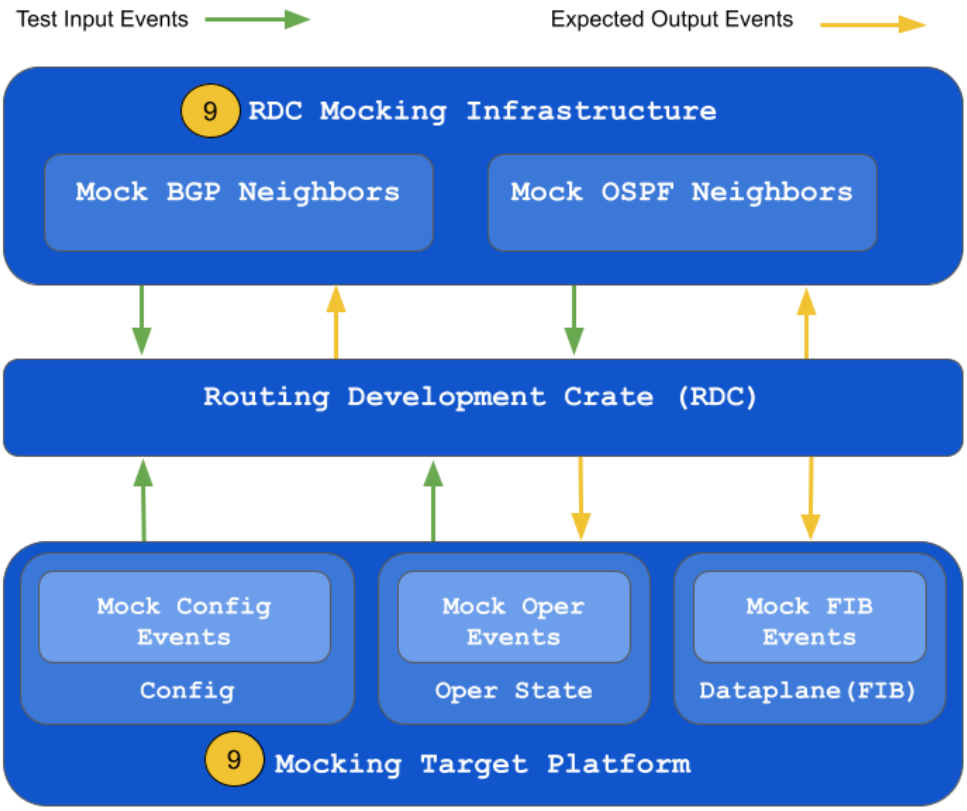


Figure 3: *RDC Testing Framework*

9. **Highly reliable**: The RDC source code includes a testing framework with hundreds of unit tests that test the internal behavior, and hundreds of integration tests that check the compliance of the external API and its conformance to RFC behavior. All these tests are run every time any single line of code is changed. This also provides a very agile development environment where changes can be comprehensively tested in seconds, rather than waiting hours for system test results. The integration tests also “fuzz test” the external API to check that malformed inputs are correctly handled and the tests track changes in memory usage.
10. **Clean code**: The RDC source code is written to be easily read by other developers. This speeds up development and also allows the compiler to implement many optimizations. The code compiles with zero warnings. Static analysis of the code also produces zero warnings.
11. **Documented code**: The RDC source code also includes a “developer guide” and a “user guide”. The “developer guide” explains the RDC architecture, how to interface with and extend the architecture, and a detailed description of the implementation of each internal component. The “user guide” is aimed at end users that are using the reference Linux IP routing suite implemented by customizing RDC. The “user guide” covers installation, configuration and operation. The contents of these guides are written in the markdown format and are included alongside the rest of the source code.

The RDC Development Team

RDC has been developed by Flock Networks ® which was founded in 2019 by Nick Carter.

Nick has over 10 years of experience designing, implementing and operating some of the largest IP networks in the world. Nick also has over 20 years experience developing IP networking equipment where he specialized in the design and implementation of IP routing protocols and IP forwarding engines.

During the initial 18 months Nick designed and implemented the first version of RDC, which included OSPF, BGP, the external API's and a comprehensive test suite. The Linux target platform implementation was also written. This became the first ever IP routing suite implemented in the Rust language.

From January 2021 onwards Nick was joined by a team of highly skilled developers to continue the development of RDC. These developers were a mix of IP protocol implementation veterans and experts in the Rust language. The IP protocol developers had a combined 160 years of industry experience developing IP routing protocols. RDC system testing was taken over by a team of highly skilled QA engineers. With the growth in team members RDC development was moved to a CI / CD pipeline with comprehensive unit test, integration test and system test coverage.

From January 2022 onwards an operations team was formed to roll out network devices built using RDC. The RDC telemetry and alarms infrastructure was finalized and put into production.

Hundreds of IP network devices built using RDC are in production, deployed across North America, EMEA and Asia. RDC has enabled these devices to be developed and deployed quickly. RDC has proved to be reliable, fast and secure in production.

Summary

Creating new network devices using out-of-date IP routing suites is a mistake. The existing implementations are insecure and do not scale well on modern multi-core hardware. The complexity involved in adding new protocols has caused the internet infrastructure to stagnate.

The internet is now a utility and the networking industry needs to advance so that the deployed network devices are so stable, secure and efficient that they become transparent to end user applications.

If you would like more information on RDC please email info@flocknetworks.com.