## Paper summary

In commodity operating systems such as Linux and windows, device drivers and extensions such as virus detectors and file systems written by less experienced programmers cause reliability issues. Poorly written extensions are majorly responsible for crashes in the system. Nooks look forward to mitigating the effect of the faulty device extensions by wrapping them in its own " lightweight kernel protection domain" called NIM. Nooks utilize existing extension architecture and use C. Nooks can prevent and recover from most extension failures and is designed for mistakes and not abuse. It provides backward compatibility, protects the kernel against driver failures, provides recovery support, and makes minimal changes to existing drivers and kernels. XPC is used for communication between the driver and the kernel and occurs between asymmetric trusted domains and Nook's object tracking code for all data transfer. The object tracking function is useful when the driver fails by providing object information for cleanup. Nooks provide tradeoffs between performance, compatibility, complexity, and completeness. Also, it can be used in third-party code as well.

## Strengths

1. The paper does an excellent job of acknowledging its drawbacks and the decision-making process.
2. Provides substantial reliability and isolation with low human efforts.
3. No need for modification to the driver code

## Weaknesses

1. The scope is confined to drivers that can be killed and restarted safely.
2. Too much overhead for intensive workloads and real-time applications.
3. Running extension in kernel mode seems like a security risk.
4. No infinite loop prevention.

## Comments for author

Potentially a great effort in improving the reliability of the system. However, only synthetic bugs are discussed. Do they depict real-world scenarios? Also is there any alternative to XPS to improve performance?