**Microsoft Makes rDSN Open Source**

Developed by System Research Group of [Microsoft Research Asia](http://www.msra.cn/zh-cn/default.aspx), Robust Distributed System Nucleus (rDSN), is now open source on [GitHub.com](https://github.com/). rDSN provides an open framework for developers, students, and researchers to quickly build and manage high-performance and robust distributed systems.

The idea of this framework arises during the team's past efforts to (semi-)automatically test, debug, optimize, operate, scale, replicate, compose, and even reason the given distributed systems . A lot of challenges were encountered in those projects, and most are due to the missing of a framework like rDSN where upper systems are built in *discipline(link to our design principles)* and the above tasks become feasible. An early version of rDSN has been used and validated in Bing for years. With feedbacks from the production teams, rDSN is improved and now made public through open sourcing, with the hope to help developers, students, and researchers who are working on distributed systems in various ways. Following are some examples for different roles (a *tutorial* is given for trying), though their usage are not restricted.

**For developers**, rDSN enhances development and management experience for system programmability, performance, and robustness. As its simplest form, rDSN can be used as an enhanced RPC library compatible to many others (e.g., Apache Thrift), or a task library where event-driven programming is adopted for high throughput. Developers can also configure rDSN into “test” mode, which tests the systems against various failures and scheduling decisions systematically, exposing early the possible bugs. Once a bug is exposed, you can switch to a “debug” mode to reproduce it. When it is online, rDSN provides automatic flow tracing and performance monitoring. If you are not satisfied with the default libraries in rDSN and want to use your own (e.g., logging or networking library), rDSN is open and you can easily integrate them. Even further, when you need to scale your service and make it reliable, rDSN replicates it with minor further development cost. In summary, rDSN provides and allows tools/frameworks to be seamlessly integrated with your system for good, and together tries to create an ecosystem as they are growing.

**For students**, rDSN provides a platform where you can easily understand and manipulate a distributed system. For instance, as rDSN adopts event-driven architecture, it generates a so-called “event matrix” which records the invocation count among named events, revealing the dependencies with weight inside the system. When learning distributed protocols, you can easily implement one atop of rDSN, and test it on its simulator. The simulator can abstract away many practical difficulties initially, and you can add them back gradually to evolve your protocol, such as from single-thread to multiple-thread, from constant message delay to variant ones, even with message lost. And there are a lot more potentials.

**Researchers** usually want to find and build something common to many distributed systems, such as runtime policies and diagnosis tools. rDSN provides a dedicated Tool API for that purpose. The API provides virtualization of all low level components, and exposes all non-deterministic behaviors from the upper applications at the event graunularity. With this API, it is much easier to build reliable and effective runtime tools and/or policies. The current release contains a handful set of examples. Even better, rDSN ensures that those tools can always be seamlessly integrated with the upper applications - a big bonus for the research work to make real impact.

With all these possible benefit, it is hoped that the community can together build better distributed systems easily, by not only adopting rDSN, but also contributing back whatever built with rDSN to help the others. Visit the project now at <https://github.com/Microsoft/rDSN>.