Even the best written software contains errors. Assistant Professor Junfeng Yang wants to unmask and correct those often subtle defects. The software bugs are costly. In 2002, the National Institute of Science and Technology put their cost at $60 billion annually. Bugs do more than crash computers. They contributed to the northeast power blackout in 2003, and delivered lethal doses of radiation to hospital patients.

“My research involves finding ways to make software more reliable,” Yang said. In graduate school, he developed an automated method to detect storage system errors. "Past tests were like throwing darts and hoping to hit a problem area. We developed systematic ways to test all possible storage states," he said.

After joining Microsoft, he extended his work to distributed storage systems on large networks. “People knew they were losing data, but not why. Our tool helped them find those bugs,” Yang said. His work led to numerous patches for Microsoft's production systems and the Linux operating system.

Now Yang is focusing on the reliability of multithreaded programs. Unlike programs that run all their instructions sequentially, multithreaded programs consist of segments, or threads, that run concurrently. Multithreaded programs are significantly faster than sequential code.

They are also more difficult to write, test, and debug. “This is because they are not deterministic,” he explained. In other words, a multithreaded program may behave somewhat differently each time it runs. “It may act correctly or buggy, depending on such variables as processor speed, operating system scheduling, and what data arrives when during operations,” Yang said.

Lack of determinism makes it difficult to reproduce errors, much less fix them. Yang’s research makes multithreaded programs execute deterministically, so programmers can isolate problems.

Explaining his approach, Yang likens threads to cars driving down a four-lane highway. “The cars drive in parallel lanes. During nondeterministic execution, they can change lanes whenever they want. When they do, sometimes they collide and cause the program to crash.

“T o make threads execute deterministically, we’ve place barriers between the lanes. We only allow threads to change lanes at fixed locations, following a fixed order. This prevents random car collisions,” he said.

Yang records this path and makes every subsequent group of cars follow it. “Because we know the path causes no collisions, there should be no collisions when another group of cars use it,” he said. By attacking multithreading, Yang hopes to weave more reliable software.

B.S., Tsinghua University (Beijing), 2000; M.S., Stanford, 2002; Ph.D., 2008

Weaving More Reliable Software

JUNFENG YANG
Assistant Professor of Computer Science