# The Space Between Theory and Practice in Distributed Systems

Marc Brooker

How do we learn synthesis? http://brooker.co.za/blog/2014/08/10/the-space-between.html

Teaching and learning about distributed systems, like any complex topic, requires real thought about what to teach and what to learn. It would be great to have enough time to teach and learn everything, but there's just too much material out there. Even if we did have the time to cover every paper, result, code base, experience report and blog post in the field, we'd still need to chose an order to cover them in. There's a natural order to things to the learned that makes them much easier to learn. That's why I was excited to see Henry Robinson's Distributed systems theory for the distributed systems engineer.

It's a good list of things to learn about, from the practical to the theoretical. I like the way it's broken up into clear sections, and uses examples from many real-world systems from industry. This is a list I'm going to be recommending to people for a while, and going to be working through myself. Unfortunately, Henry's list reflects a greater gap in the overall literature: the gap between theory and practice.

From Dynamo and Cassandra to Chubby and ZooKeeper there's a wealth of content available on the design and implementation of real systems. Some of these papers go into real depth on seemingly small details (like Paxos Made Live) while others concern themselves with high-level architecture. Combined with things like Deutsch's 8 Fallacies and Jeff Hodges Notes on Distributed Systems for Young Bloods there's a lot of practical advice available to learn about the practical side of distributed systems.

On the theoretical side, there's also a wealth of material. Robinson points to the CAP proof and the FLP result, and admits that he's only just scratching the surface. There are thousands of good theoretical results out there, from the usual suspects like Lamport and Lynch to areas like topology and game theory.

I feel like if I went through everything I've read on distributed systems and arranged them on a spectrum from *theory* to *practice* the two ends would be really well populated, but the middle would be disturbingly empty. Worse, changing to a graph of citation links would show a low density from theory to practice. I strongly believe that a deep knowledge of theory makes practitioners smarter and better. I believe that a deep knowledge of practice makes researcher's work more relevant. It would be great to see more material in this gap.

Many will point at this stage that it's not a complete gap. I'll admit that there's some great material there, including Paxos Made Live on the theory end of practice and Kenneth Birman's Guide to Reliable Distributed Systems or Butler Lampson's How to Build a Highly Available System Using Consensus on the practice end of theory. There are also blogs like Henry's and Aphyr's which do a good job in that gap. Despite this, I still see some big gaps in material. An example may be the easiest way to illustrate it:

> If FLP says consensus is impossible with one faulty process, and faults happen all the time in practice, how are real systems built with consensus?

There are a few ways to answer this question. One starts with pointing out that FLP talks about it being *not always possible* to solve consensus, rather than *never possible*. Another way is to point out that the real world is richer than FLP's idealized model, and the problem can be solved with clocks or a random oracle. A third way is to laugh derisively at the asker and point out that the answer is in Paxos Made Simple (feigning surprise *What? You haven't read Paxos Made Simple?*).

Despite these 'obvious' answers, it's actually a really interesting question. On one side we see a researcher saying that consensus isn't always possible, and on the other we hear practitioners talking about how they built highly-available systems using consensus algorithms. Who is right? Does the researcher have their head too far in the clouds? Is the practitioner so ignorant of theory that they have built a ticking time bomb?

That's the gap I am talking about: material that explains how the practice is synthesized from the theory, and how the theory is based off analysis of the practice. The exercise of synthesis is very seldom straight forward, but we too frequently leave it to the imagination. In this context, I use *synthesis* to mean the process of gathering ideas from the literature and putting them together into a whole working system. Related processes include analysis of other systems, where we break them down into their consistuent parts and see what makes them work (or not work). These are among the most important processes behind successful engineering, but are written about least.

I would love to see more material focused on exactly this synthesis problem in distributed systems, because I think it would help improve the quality of practice, and strengthen the dialog between practitioners and researchers. That's good for all of us.