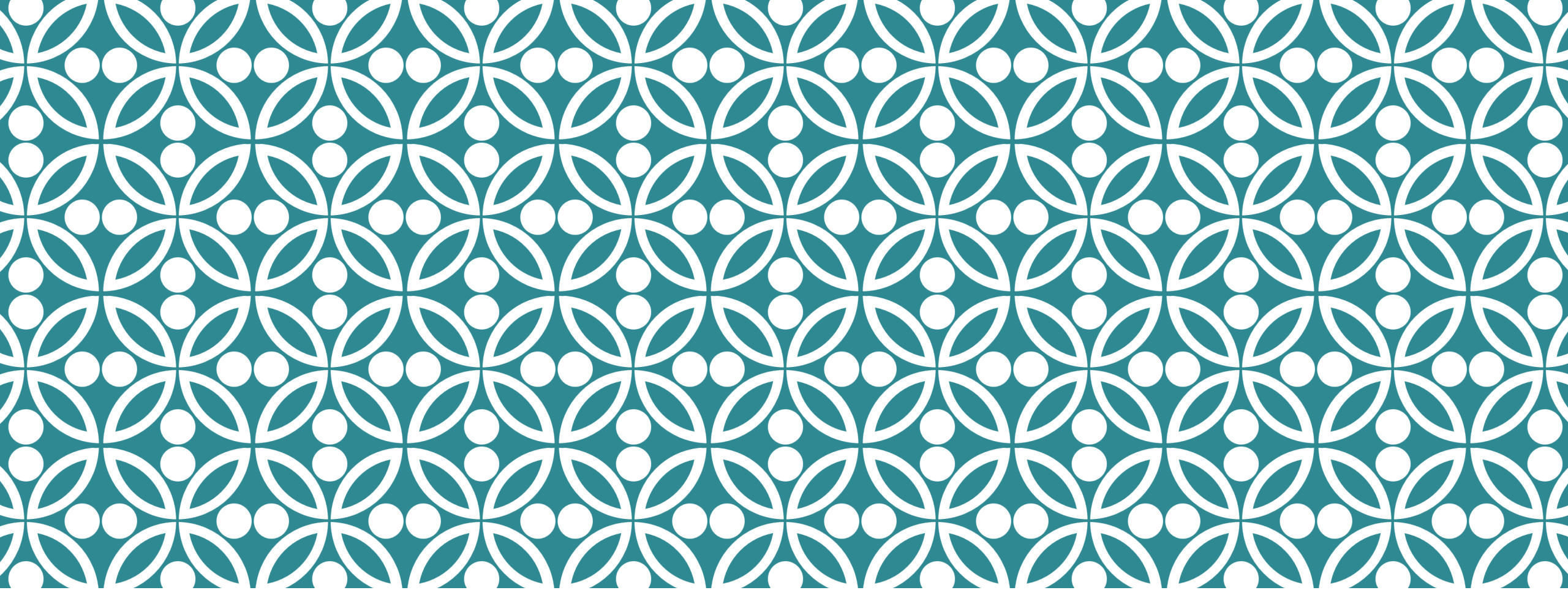


VALIDATING HIGH LEVEL SIMULATION RESULTS AGAINST EXPERIMENTAL DATA AND LOW LEVEL SIMULATION: A CASE STUDY

David Griffin, James Harbin, Alan Burns, Iain Bate, Robert I. Davis, Leandro Soares Indrusiak

WIRELESS SENSOR NODES

For when you want to transmit data around a place without wires

Reduces costs

Improve reliability, or at least trade one set of reliability concerns for another

Makes it much easier to deploy if some of your nodes might move

MIXED CRITICALITY SYSTEMS ON WIRELESS SENSOR NODES COMMUNICATIONS

Intuition: Rather than have a definitively known worst case execution time for each task, have more than one!

Most current work uses two: C_{Lo} and C_{Hi}

If a task exceeds a criticality budget, the system gracefully degrades by reallocating resources to more important tasks

- E.g. if a task exceeds its C_{lo}, all low criticality tasks get their resources lowered
- The protocol might allow for online recovery, or it might not

Attractive for wireless communications: Wireless communications are a scarce resource shared by all nodes in range

AIRTIGHT

Protocol for Mixed Criticality Wireless Sensor Networks Communication [Burns et al. 2018]

Synchronises clocks of nodes and uses a precomputed global table to dictate which nodes should send/receive at any time

Receiving nodes send an ACK packet

If a sending node doesn't receive an ACK for a packet, it will try to resend that packet

Tries harder to resend high-criticality packets than low-criticality messages

If failures get too frequent, AirTight stops sending low-criticality messages until the situation improves

IT'S NOT JUST...

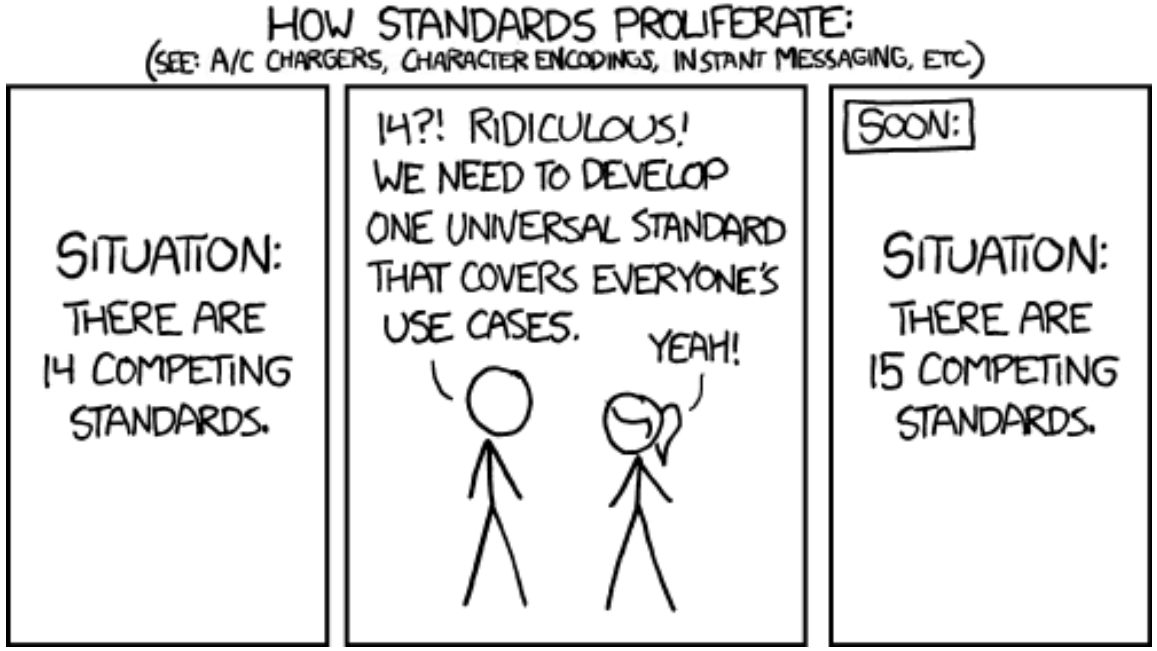
AirTight is very lightweight and is designed to be easy to analyze, which isn't the case for most competitors

- e.g. most difficult stuff is computed offline

This paper uses a low-level custom implementation of AirTight, but AirTight isn't an inherently low-level protocol. It will work on top of more common standards.

- Just needs the ability to be able to send data and measure faults

Image credit: https://xkcd.com/927/



EXPERIMENTING WITH WIRELESS SENSOR NETWORKS

It's a pain.

Nodes have very little storage, so we can't actually get most of the interesting data from them

- Unless we wire them into a computer, defeating the point of "***wireless*** sensor networks" as well as introducing probe effects

Nodes are also slow, so experiments on real hardware take ages

It's impossible to control all the variables, like background noise and therefore transmission errors

Setting up experiments with real hardware is difficult

This makes it hard to develop a new wireless sensor network protocol using just real world experiments

HIGH LEVEL SIMULATORS

High level simulators provide a way around these issues

Work much faster, and we can see everything that's happening

York implemented a good high level simulator for AirTight

Abstracts away a lot of messy details of the real world

Gives results which are pretty close to the real world

HIGH LEVEL SIMULATORS

But how do we know that the results are reasonable?

Simulator and Real-world results obviously differ because the simulator doesn't do a lot of things

- Simulator models a constant failure rate, but in real life failure rates can change over time
  - And published simulator experiments assume failure rate = 0 unless we're explicitly testing fault handling
- Simulator assumes that an ACK always arrives
- Simulator assumes perfect clock synchronisation
- Simulator assumes packets are an indivisible unit

Also simulator sometimes logs results which are curious

- When you look at the detail, there are some noticeable differences

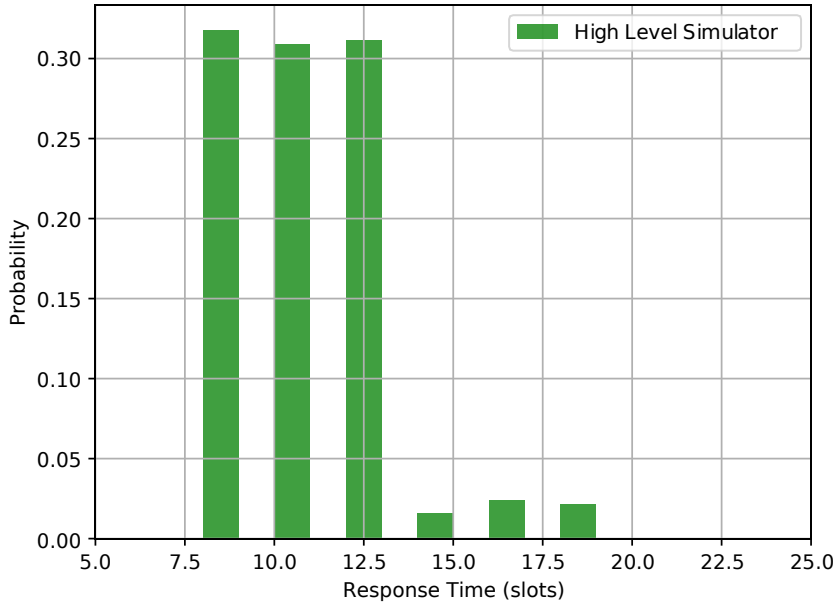
DIFFERENCES BETWEEN REAL-WORLD AND HIGH-LEVEL SIMULATOR

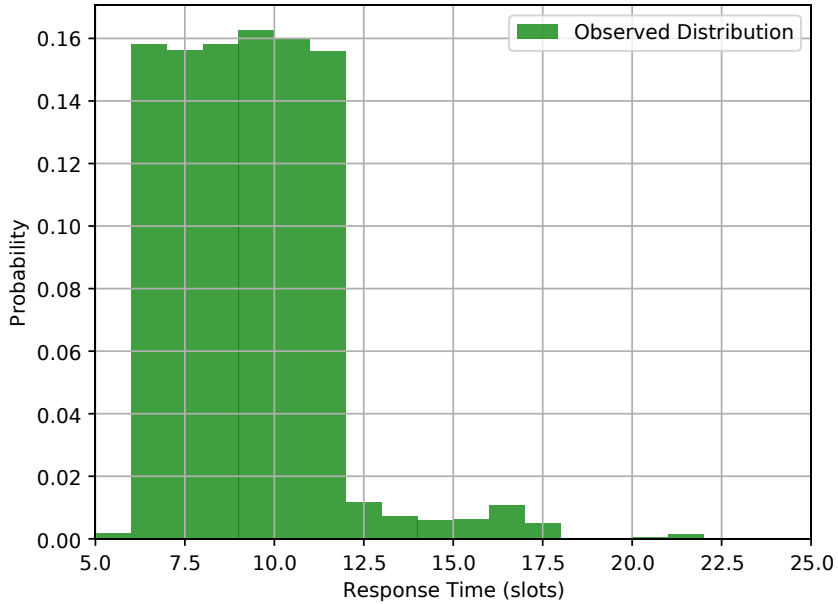
These differences aren't that *bad*

Almost all the differences are a 1-slot pessimism in the high-level simulator

It just seems odd that the high-level simulator exhibits a pattern not seen in the real-world experiment

This begs the question: Can we really trust the high-level simulator?





WHAT TO DO ABOUT THE DIFFERENCES?

Could treat these as bugs to fix in the high-level simulator

- But that could introduce more complexity to the high-level simulator, making it slower and harder to understand

Could devise experiments to determine exact cause of differences

- But that requires doing more experiments on real hardware, and getting all the data off the real hardware is tricky

What we want is a way to characterise the difference between already existing experiments *without* having to do more work with real hardware.

That means we have to represent *both* of the real-world and high-level simulator experiments in one framework.

SOLUTION: LOW-LEVEL SIMULATOR

Implement a low-level simulator which is capable of modelling the real world more accurately *as well as the high-level simulator*

- Works at a sub-packet level
- Models failure of ACKs
- Models clock drift
- User can see everything that's happening

Not a replacement for the high-level simulator because it's far too slow

But creates a system where the user controls *all* variables

SOLUTION: LOW-LEVEL SIMULATOR

Create *two* configurations for the low-level simulator

1. Match assumptions with the high-level simulator
   - Assume ACKs never fail, constant failure rates etc.
2. Calibrated against real-world experiments
   - ACKs can fail, failure rate models data extracted from experiments, clock synchronisation can fail

SOLUTION: LOW-LEVEL SIMULATOR

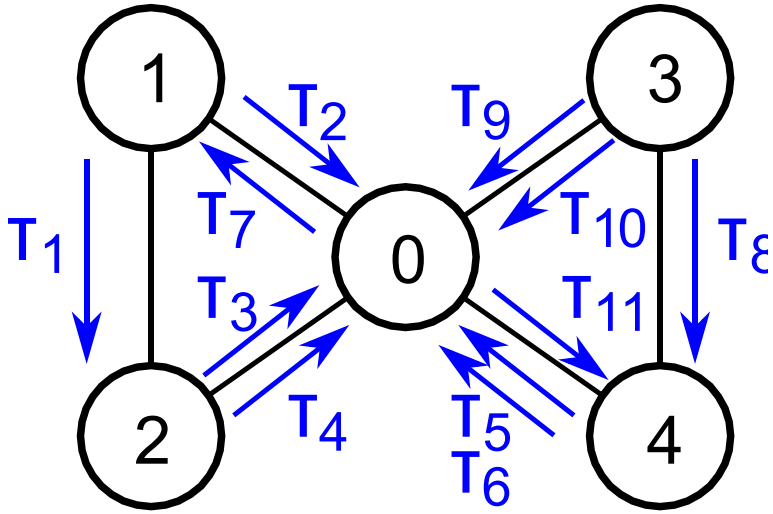
Can then compare

- High-level simulator to the Low-level simulator (matching the High-level simulator assumptions)
- Real world experiments to the Low-level simulator (matching real world instrumentation)

If both of these are good matches, then differences between the High-level simulator and the Real world experiments can be explained by the difference in configuration of the Low-level simulator

- Which can then be fed into development of the High-level simulator to improve its accuracy

EXPERIMENT



Uses the data from the small-scale experiment in Burns et al. 2018

Specification:

5 Nodes, 5 High Criticality Flows, 6 Low Criticality Flows

- Full details in paper
- Each flow was analyzed separately, using appropriate statistical tests to determine whether results matched

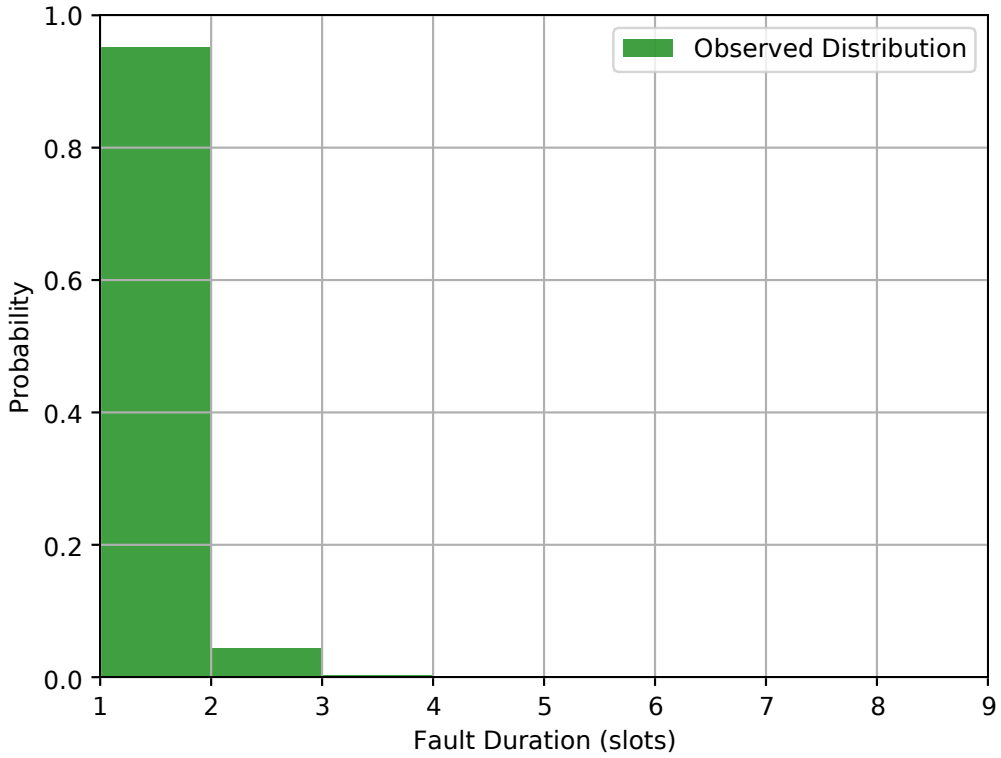
CALIBRATING THE LOW LEVEL SIMULATOR: EXAMINING THE REAL-WORLD

Most of the faults are of single slot duration

In fact, this distribution is exactly what you'd see if every fault had a duration of < 1 slot

The faults of length 2 are likely two separate faults that happened near each other

(There exists one fault of duration 8 which is not statistically significant)

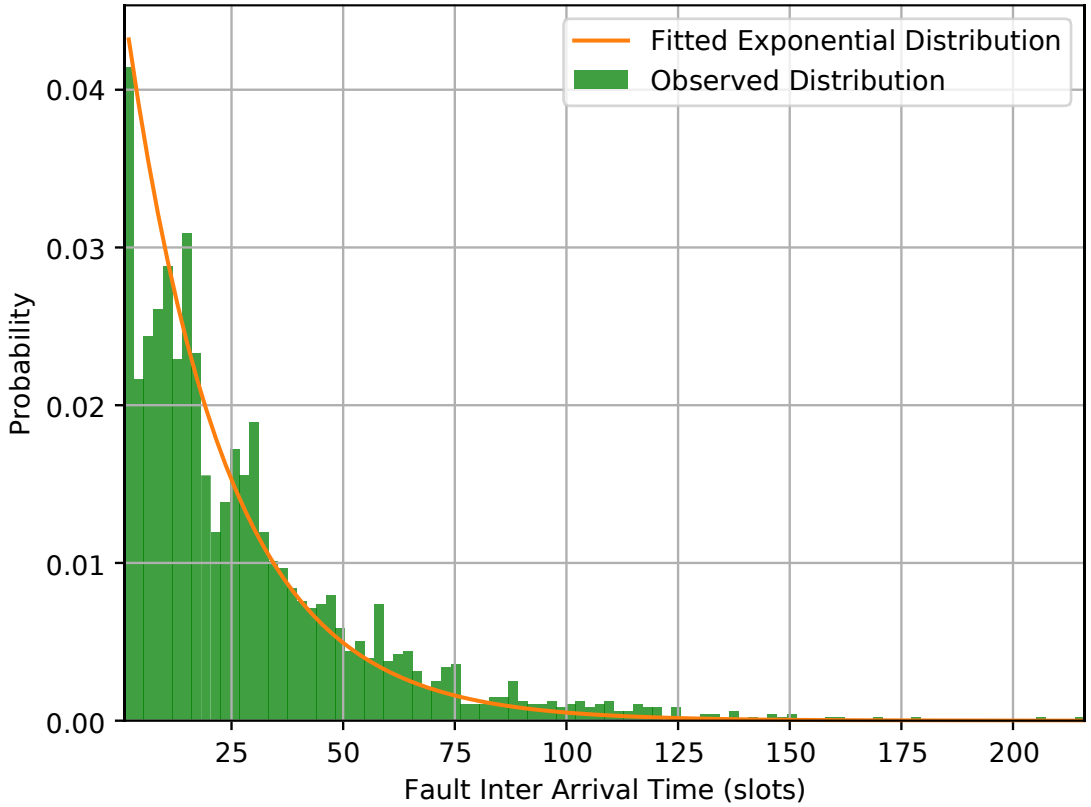


CALIBRATING THE LOW LEVEL SIMULATOR: EXAMINING THE REAL-WORLD

And when you look at the fault inter-arrival times, they look pretty much exactly like a random distribution of point faults

So the real-world experiments tell us we likely have faults occurring at a sub-slot level, because the phenomena behaves like an instantaneous event

This goes against the design of the high-level simulator, which was designed to test AirTight's resilience to long lasting faults

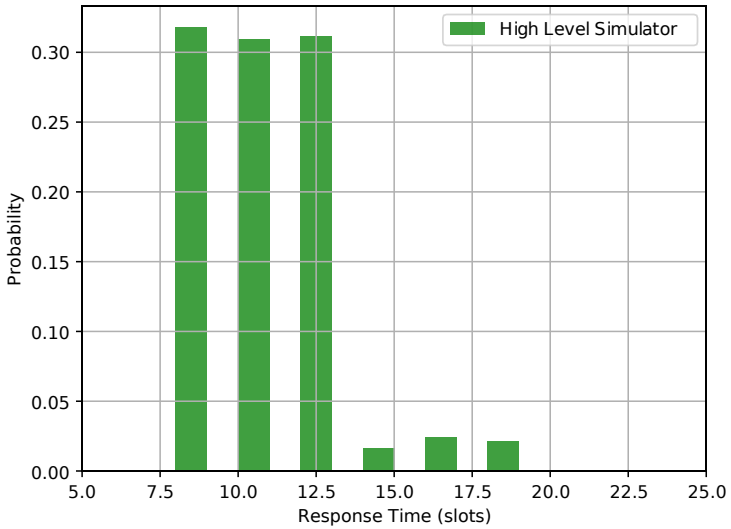


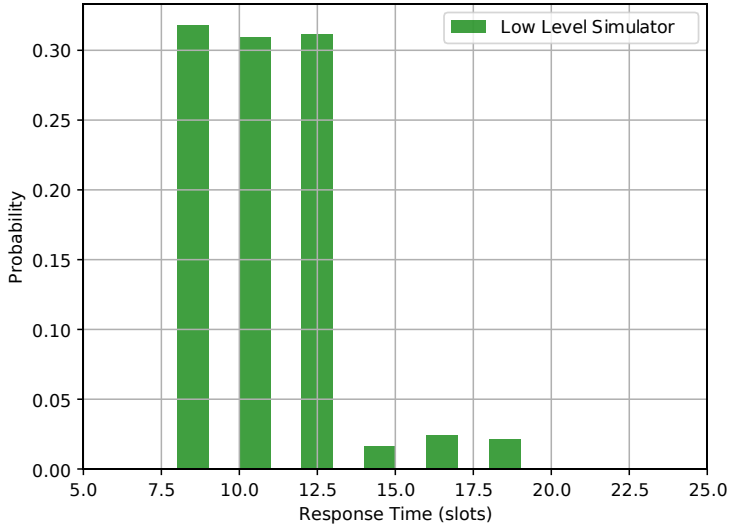
HIGH LEVEL VS LOW LEVEL SIMULATOR

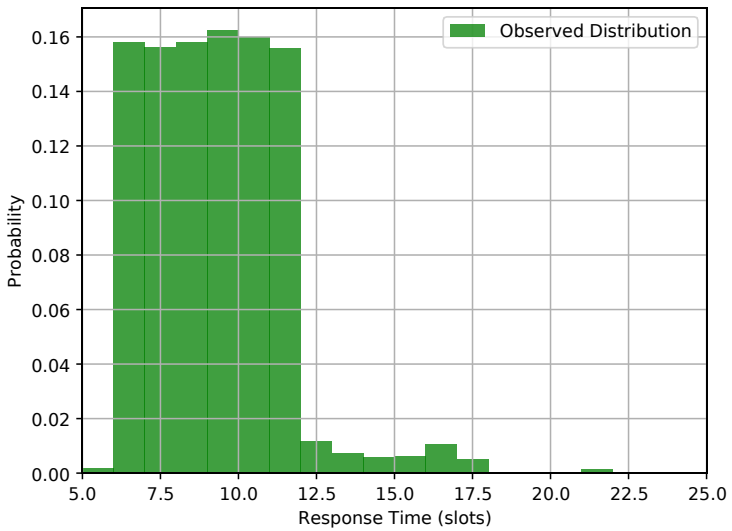
As can be seen, the low-level simulator can match the high-level simulator perfectly

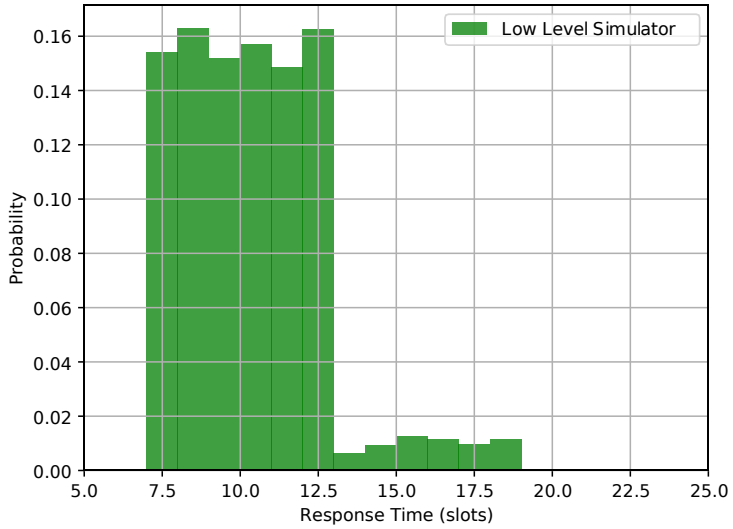
This is accomplished by

- Assuming all ACKs delivered
- Assuming clock synchronization
- Assuming constant failure rate









REAL WORLD EXPERIMENTS VS LOW LEVEL SIMULATOR

Very similar, but not identical as experiments are obviously not able to replicate the exact conditions of the real world

Nodes have a randomized (but small) amount of clock drift

- This would have been noticed earlier if our experiment sent payload data
- In this experiment nodes are almost always listening, meaning the system is tolerant to clock-drift

Fault rate is constant, but modelled on the real-data rather than set

ACK packets can fail at a rate derived from the data

CONCLUSIONS ON THE EXPERIMENT

The High-level simulator is accurate in most respects, and much faster than any competitor

- Speed is important for a lot of experiments in design!

Low-level simulator can get really close to both the results of the high level simulator and the real world experiments

The experiments suggests that *Clock Drift*, *ACK packet failure* are our most likely culprits for discrepancies between High-level simulator and reality

Clock Drift could be an issue when it comes to efficiency in the real world

- Can't have two nodes transmitting at same time
- Have to pad slots to accommodate clock drift, or synchronise better, or use better nodes

CONCLUSIONS ON THE METHOD

Given a sufficiently configurable low-level simulator, *simulator configuration* is a good way of characterising how sets of results differ

This allows a concise and easy to understand explanation of differences between a high-level simulation and real-world experiments

These differences are useful to understand as they can indicate whether or not the difference in the results is manageable, or indicates a more fundamental problem that needs attention

Hence explaining this difference is important to have confidence that the results of the high-level simulator are fit for purpose