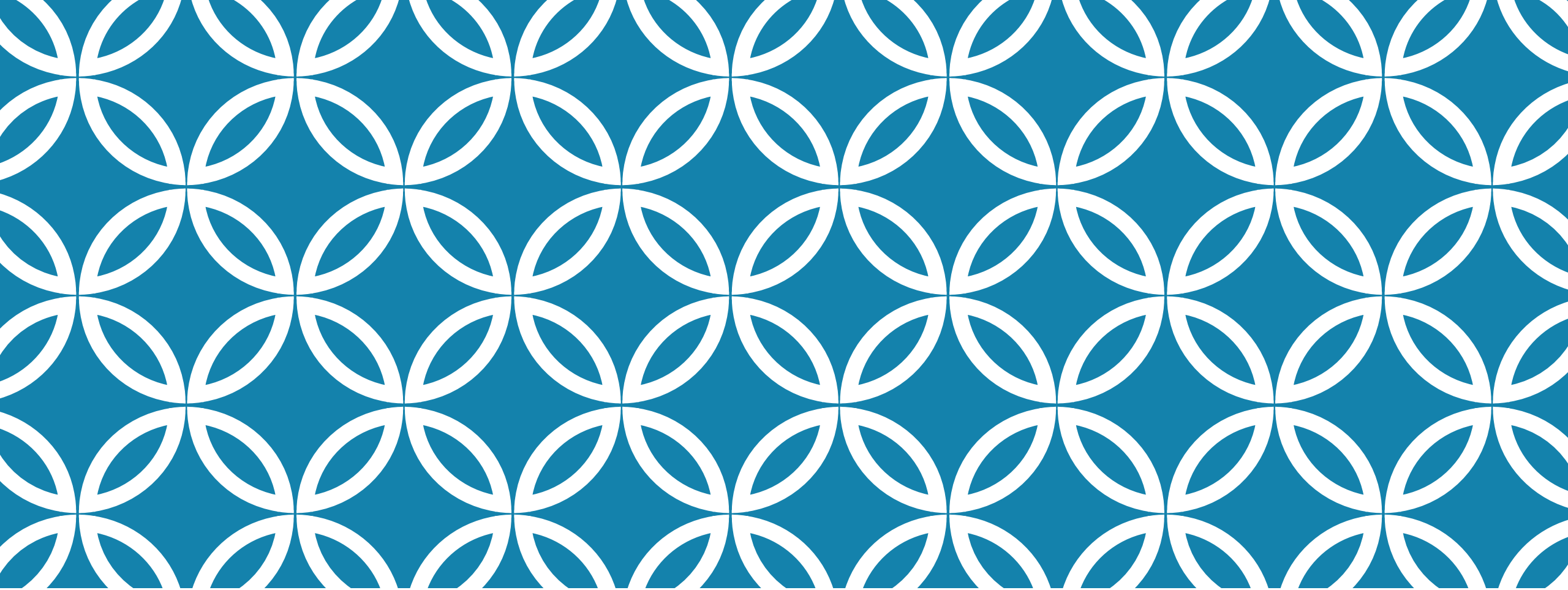

CS5412 PROJECT INFORMATION

Ken Birman

CS5412 Spring 2019

PROJECT GOALS

We are hoping you will:

- Gain hands-on experience on Microsoft Azure IoT + Azure Intelligent Edge. AWS would be acceptable too, but we prefer Azure.
- Learn to work in a team with varied skills and specializations.
- Solve a real "digital farming" application, using real sensors, real data, real machine learning tools, and working with a real "customer".
- Use existing infrastructure where available (like Azure functions, Azure message queues, etc).
- If you need to create a new microservice, work with Cornell's Derecho library.
  * Use Derecho's Ubuntu VM. Ubuntu is a version of Linux that works well on Azure.
  * Derecho's strongly consistent key-value "object store" is the obvious place to start.

FORMING TEAMS

Projects are done in groups of 2-3 CS5412 students.

Many will have one more "external consultant:" a student from the college of agriculture and life sciences who works in farming, agriculture or a related topic and is taking a course with a similar project need.

The jointly done project would satisfy requirements in both classes.

MEETING PEOPLE


© Can Stock Photo

We'll have a few "meet and match" events in the first weeks.

Many projects are done by people who only know each other in passing. But this is true in industry as well! Don't focus on "finding friends."

The best teams often bring people together with very different specializations because doing so gives you coverage of more aspects.

ARE THERE "PURE CLOUD" PROJECTS TOO?

Yes, a few groups will probably work on creating technology for the cloud and won't partner with people from CALS. They would bring us ideas for our approval. *All projects must have a significant IoT-cloud dimension.*

A few groups will use CALS data but won't have a CALS participant.

But we are aiming for as many joint projects as feasible.

WHY ARE WE FOCUSED ON AZURE?

Microsoft is leading among companies with high quality, professional products for cloud + IoT, so this drives a focus on their platform.

If you love AWS for some reason, you can do similar things, but will often find yourself looking to see how Microsoft did it, then trying to copy them.

So we will emphasize Azure, used from Linux (not Windows).

WHY NOT AWS OR GOOGLE?

Actually, we don't mind if you prefer to use some other cloud, but we are trying to offer some concrete help, and we can't help with three cloud options all at once.

These days Azure is just using a standard Ubuntu Linux as its default and we can create prebuilt containers for you to run with the software we want you to use already installed.

You can still code in whatever language you prefer.

WHAT ABOUT PROGRAMMING LANGUAGES?

These days, C++ 17 is probably the first choice for IoT application development, among people who care about performance.

But you can work in any language you like, at a slight cost.

For our class, we won't tell you what language to use We hope to have a Java JNI interface for the object store by the time you would need it, so Java would then also be an option (perhaps Python, too).

HOW TO LEARN AZURE?

As it turns out, Azure is "recipe-oriented":

- There is a standard way to build scalable applications, and there is a form-fill style of programming to match this standard approach.
- You can create new μ-services, but it takes a bit more sophistication.
- Only the recommended styles of programming will give good outcomes.
- Experts can do anything, but nobody is an expert after six weeks!

Sounds, nasty, right?

VISUAL STUDIO 2017 AND VS CODE

We use cloud-oriented interactive development environments (IDEs).

For Azure, Visual Studio 2017 or a related editor called Visual Studio Code are best (VS Code is a good choice if you favor C++ in one of the Azure/Linux options, while Visual Studio is best for the Windows .NET)

In both editors, you can edit, compile, run, debug, profile, etc. But the key thing is that both have built-in templates for common patterns!

EXAMPLES?

In Visual Studio 2017, you can just tell it to create a C# program that will:

- … accept RPC requests over the Internet (via Windows Comm. Framework, WCF, which can use a built-in message format, or SOAP/REST)
- … create a client program that can issue those RPC requests.
- … implement a function to run in the Azure IoT Edge Function Server, or the Azure Intelligent Edge for IoT.
- … etc (the list is very long)

Using these tools, they provide the template and you just fill in the blanks!

CAN I USE JAVA+ECLIPSE?

Yes, but it might be harder.

Microsoft and Amazon each favor their own IDEs, as noted (Cloud 9" is the preferred IDE option on Amazon AWS). Both already understand cloud programming.

With Eclipse you might have to find and install the relevant "templates" more or less one by one, and this takes more expertise. But if you prefer this approach you can find instructions easily on the web.

STEPS TO DOING A TYPICAL PROJECT

Meet with your team members and agree on project goals. Some groups will chose to define the details on their own! In industry, consulting teams often are asked to look at a company "need" and then to recommend ideas.

So this role of looking at an area and then coming up with a project is real.

But for joint projects with CALS students we will provide structure including goal, data, and even some fairly concrete recommendations.

EVERY PROJECT MUST HAVE AN IoT FOCUS!

Unless you are proposing a new IoT infrastructure idea, this means:

- Identify data sources and sensors (like drones + still images, or video).
- Have access to a real data set, or a plan to create real data in which case you must have your own data sources and sensors planned.
- Have a concrete goal. This could be just data visualization, but would often include some form of data analytics, machine learning, database creation, cloud-directed "actions", etc.
- Have a structure closely matched to standard recipes, using roll-your-own solutions only where you genuinely push beyond what is already available.

PROJECTS MUST ALSO BE "FREE"

You should get free accounts on Microsoft Azure or Amazon AWS.

You should use our sensors+data or *have a plan of your own* to procure data or sensors. *In 2019 we have no funding we can provide to help.*

Your software will have been created in part using Cornell resources and hence should follow Cornell's recommendations on IP: It should be open source, under the Apache permissive or 3-clause BSD licensing.

BUT I NEED MY CODE FOR MY STARTUP!

Cornell is very keen to encourage entrepreneurship! There is no restriction at all on using work done in CS5412 for a startup later. Several past students were successful doing that!

Open source also means "open for you, down the road".

Cornell will even help you find incubator space and funding. But the IP model for work done as a student using (even partially) Cornell resources is dictated by public law and Cornell policy.

STEPS TO DOING A TYPICAL PROJECT

You need to think about

- Sensors, how to register them with Azure IoT Hub, configure them.
- Data: what to download, how to decide what to process and what to store or discard, how to manage limited space on the sensors themselves
- The data processing pipeline. Machine learning: roles it might play, how you will train the model, where the model will be hosted, how it will be updated.
- Output you will create, or actions your solution will take, and how.

MAKE A BLOCK DIAGRAM!

Look at some of the pre-built recipes on Azure.

They always have block diagrams. Find one that seems like a good match and then customize it to be a perfect match to your goals.

Need to code new Azure functions? No problem, but learn how to build "hello world" before you get really fancy!

Need a new μ-service? Derecho can help you build it. Try to base it on the object store (we will learn all about it in one of our first lectures).

I NEED RDMA! AND OPTANE MEMORY!

Cutting edge hardware is cool. Still, for these demonstrations we are hoping people won't need fancy hardware accelerators.

But if needed, for convincing reasons, we can provide access to Cornell clusters with the same versions of Ubuntu found on Azure, and with high speed memory and RDMA.

Since Derecho works perfectly well on TCP, you may need a *very* convincing reason to get permission to do this. Why not build on TCP first?

HELP! I'M STUCK!

We have a crew of insanely amazing TAs and other helpers.

If you get stuck, come talk to us. We'll get your group rebooted!

It helps a lot to start with an Azure "recipe" and its sample code, and then slowly modify it, rebuild, try again. Small steps.

HELP! MY TEAMMATE IS A JERK!

This happens sometimes. Plus people do travel for interviews and can drop the ball.

We allow groups to fragment and recombine if needed, but we hope you can try and work things out on your own first.

Even for groups in industry, “stuff happens.”