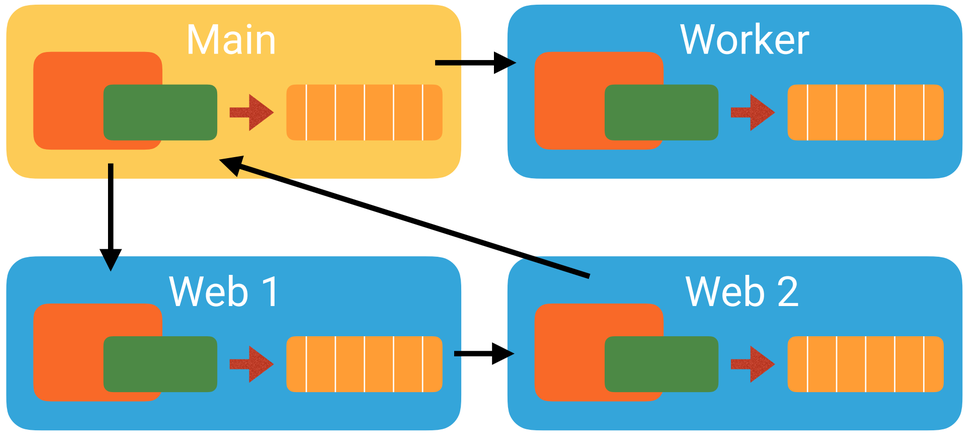
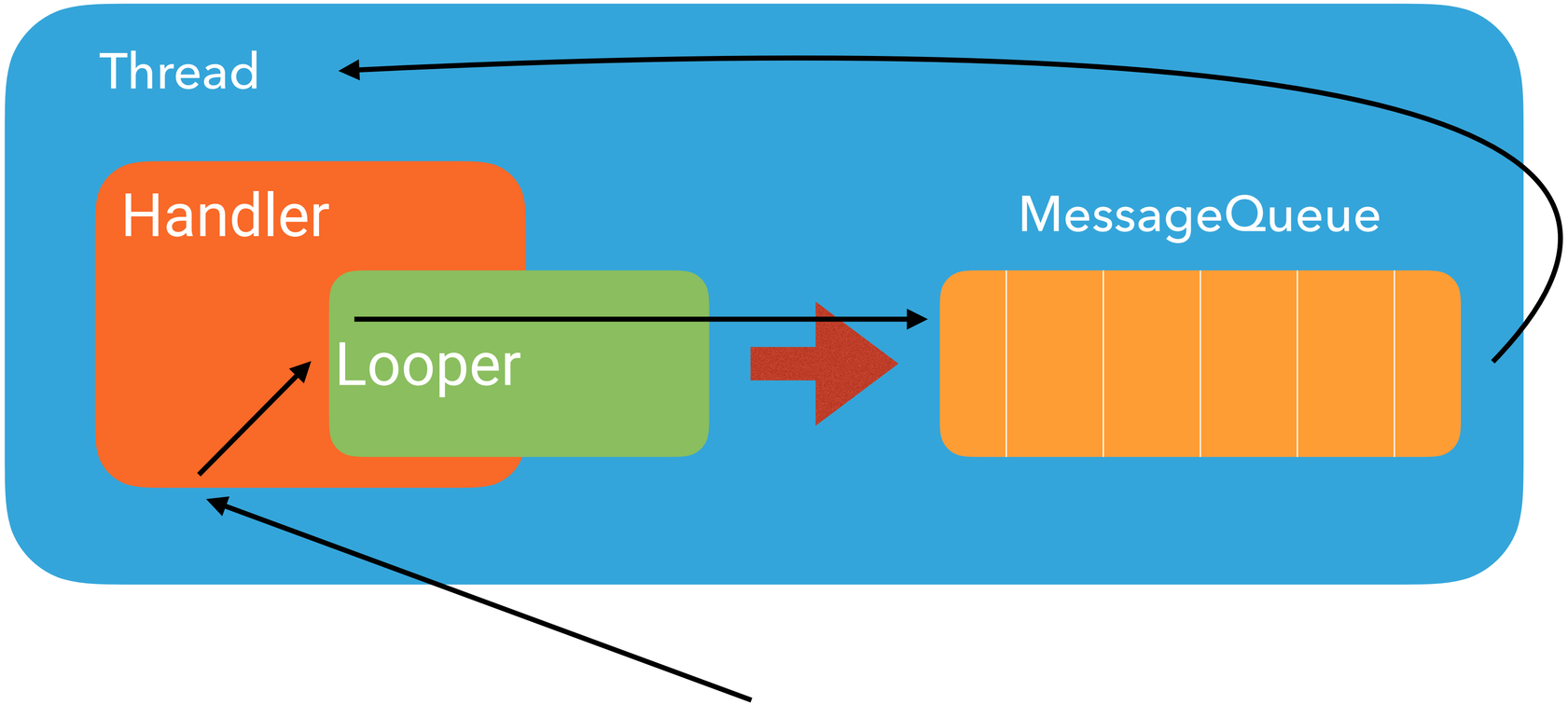


MOBILE APPLICATION DEVELOPMENT

ANDROID (2017)

LECTURE 19: NETWORK COMMUNICATION

THREADS AND HANDLERS



ASYNCHRONOUS NETWORKING OPERATIONS

- Network operations on the main thread are (usually) prohibited on Android.
- In order to carry out network requests properly, asynchronous operations must be used to perform networking and the UI must then be updated on the main thread.
  - Not doing this will usually cause an Exception to be thrown.
  - There are multiple methods of accomplishing this sort of operation, all of which make use of multiple threads.
- The example we will cover for this class uses the AsyncTask class, which is a high-level abstraction designed to simplify asynchronous calls.

ASYNCTASK

- AsyncTask is a wrapper class for a number of Thread and Handler functions, which allows work to be done off the main thread but for the main thread to be updated easily as the work on the background thread progresses and finishes.
- Provides a set of overridable methods to handle certain events:
  - Use `onPreExecute()` to perform setup (on main thread) and prepare to run the task.
  - Use `doInBackground()` to specify work that is done off the main thread.
  - Use `onProgressUpdate()` to update the main thread with the task's progress.
  - Use `onPostExecute()` to update the main thread when the task is finished.
- The parameters to and return values from these methods are defined by the programmer.

DEFINING AN ASYNCTASK

- AsyncTask declarations take the form AsyncTask<Params, Progress, Results>, which allows flexibility for how the various methods in the class operate.
- Params, Progress, and Results types affect the AsyncTask implementation.
  - Params determines the type of the varargs passed to `doInBackground()`.
  - Progress determines the type of the varargs passed to `onProgressUpdate()`.
  - Results determines the type of the varargs passed to `onPostExecute()`. These varargs are the return value(s) of `doInBackground()`.
- AsyncTasks may also use internal variables to maintain state beyond these defined parameters and return types.

NETWORKING BEST PRACTICES

- Always check status codes, return values, etc. to validate that everything worked.
- Avoid polling servers or rapidly retrying failed network calls if at all possible.
- If something CAN happen on a background thread, it probably should.
- Use locks to protect shared data structures, but share as little as possible.
- Show the user progress indications if network tasks are expected to take a while.
- Use the most secure connection you have available, do not send sensitive data about the user over an unencrypted connection.
- Don't use the network more than you need to.