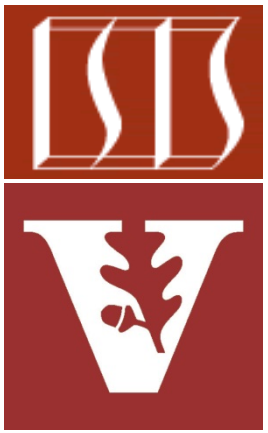
Android Concurrency: Evaluation Android's Concurrency Frameworks

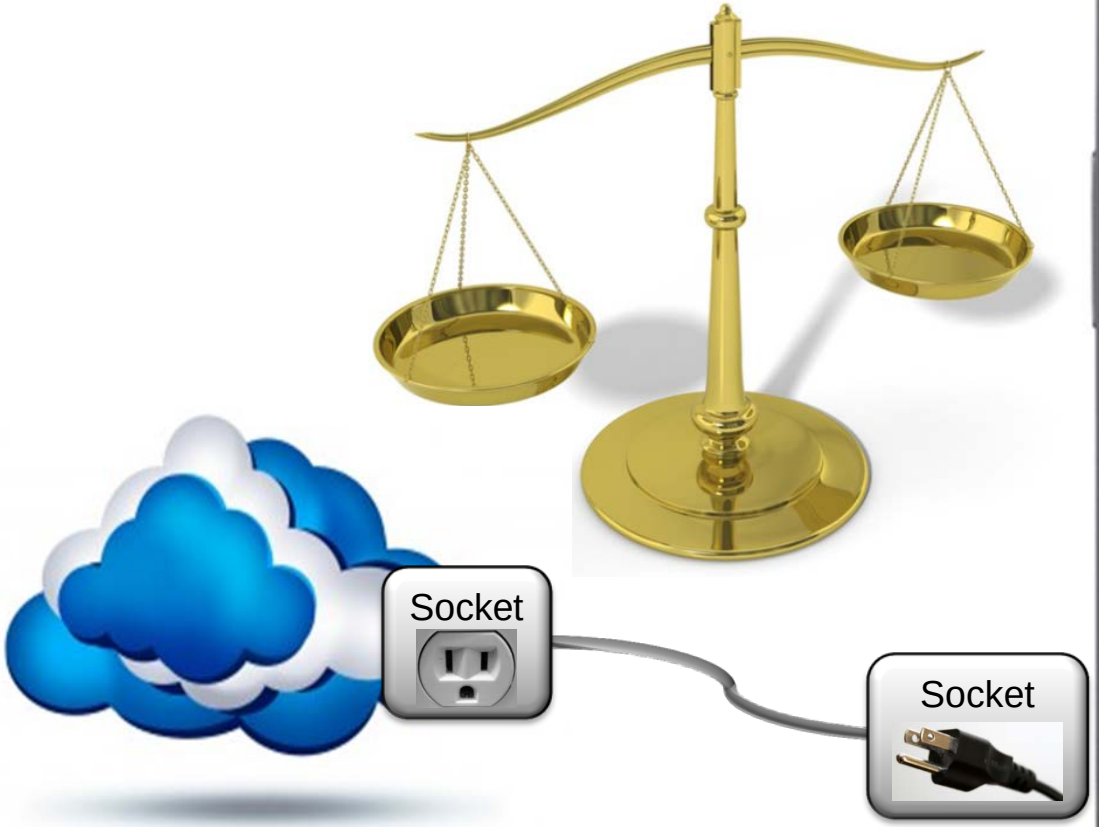


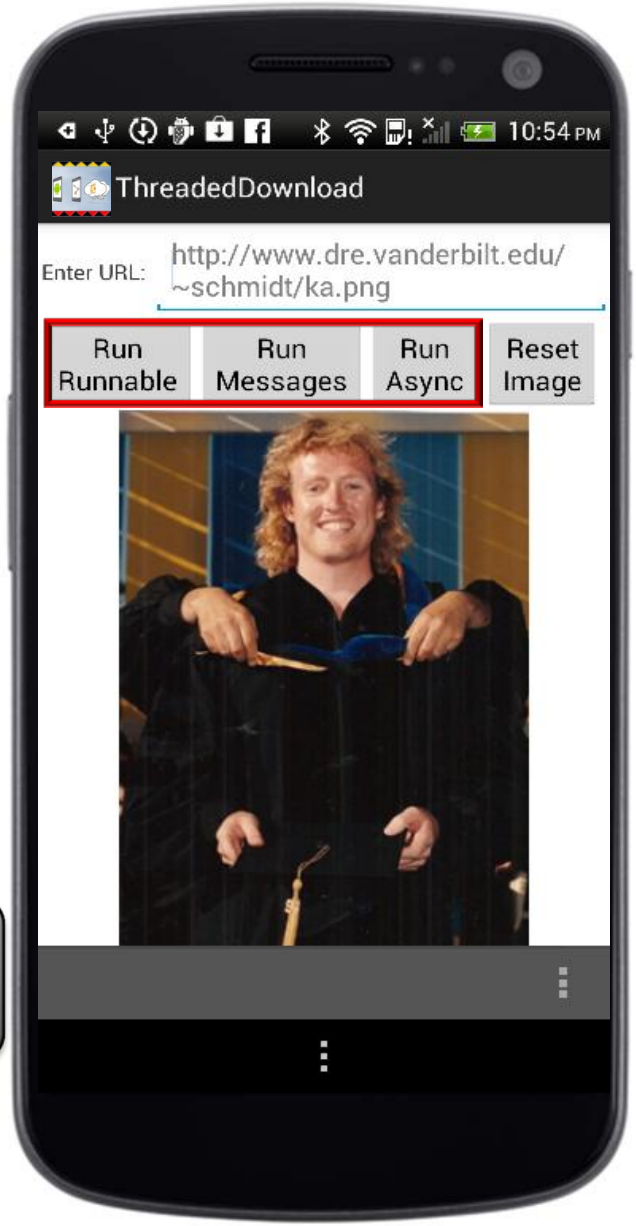
Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software
Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Module

- Understand how to evaluate the Android concurrency frameworks along several dimensions to select the appropriate framework for your applications





Evaluating ThreadedDownloads Solutions (Part 1)

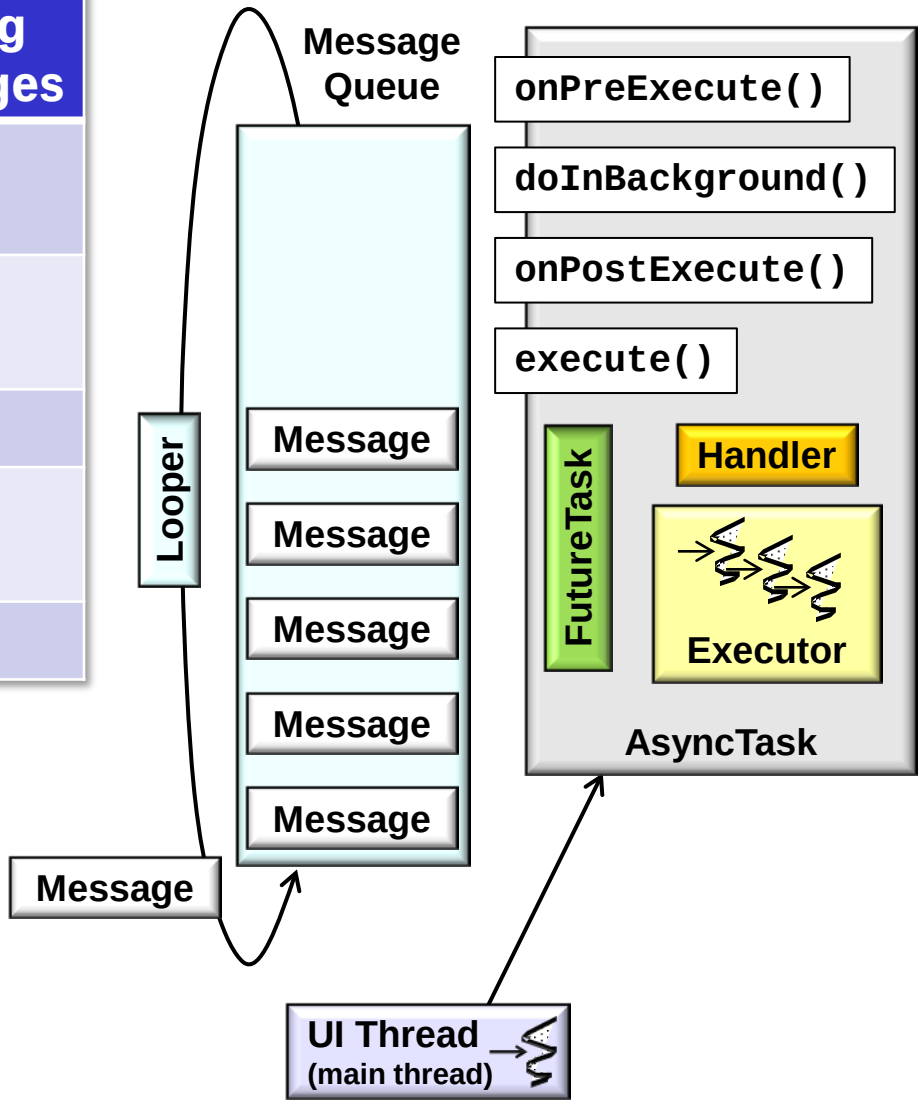
Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	□□□	□□□	□□
Usability (Complex)	□□□	□	□□
Scalability	□□□	□	□
Flexibility	□□	□	□□□
Efficiency	□□	□□□	□□□

Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	□□□	□□□	□□
Usability (Complex)	□□□	□	□□
Scalability	□□□	□	□
Flexibility	□□	□	□□□
Efficiency	□□	□□□	□□□

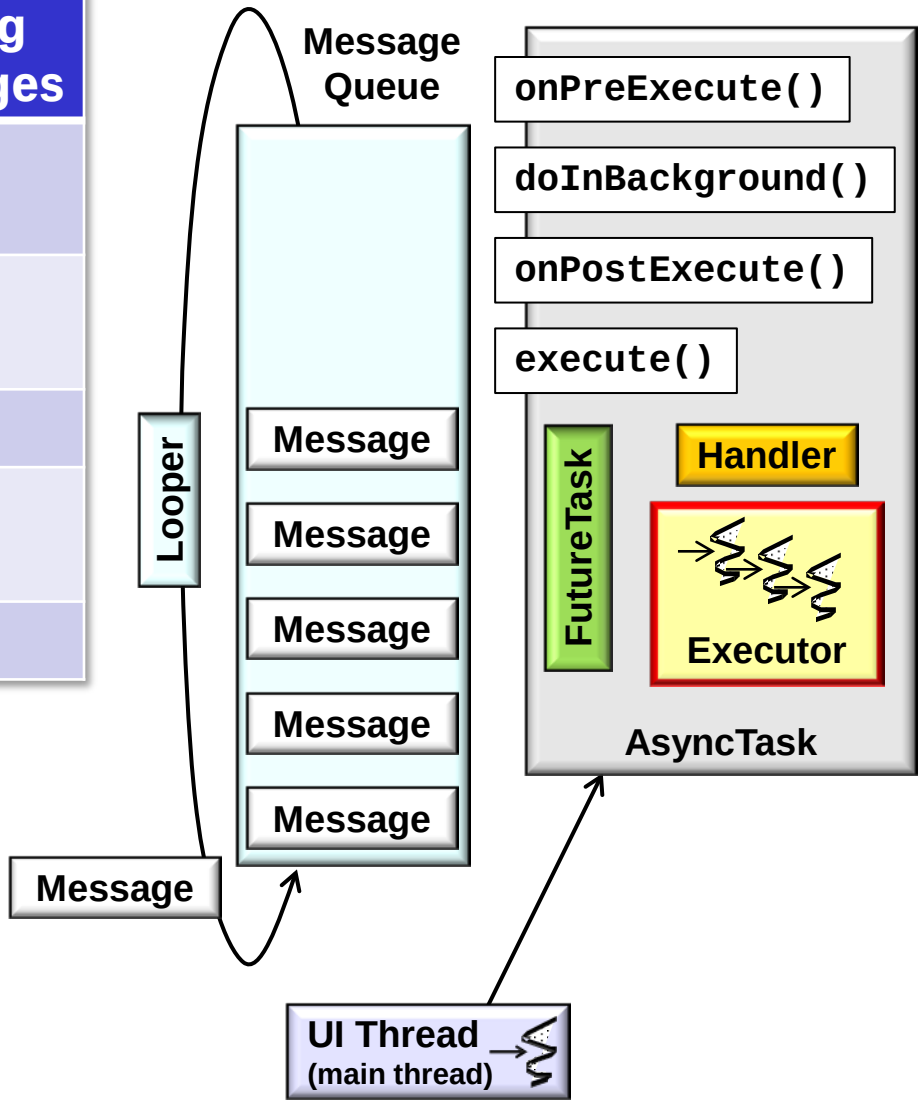
- Simplifies development of both simply & more complicated concurrent applications



Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	□□□	□□□	□□
Usability (Complex)	□□□	□	□□
Scalability	□□□	□	□
Flexibility	□□	□	□□□
Efficiency	□□	□□□	□□□

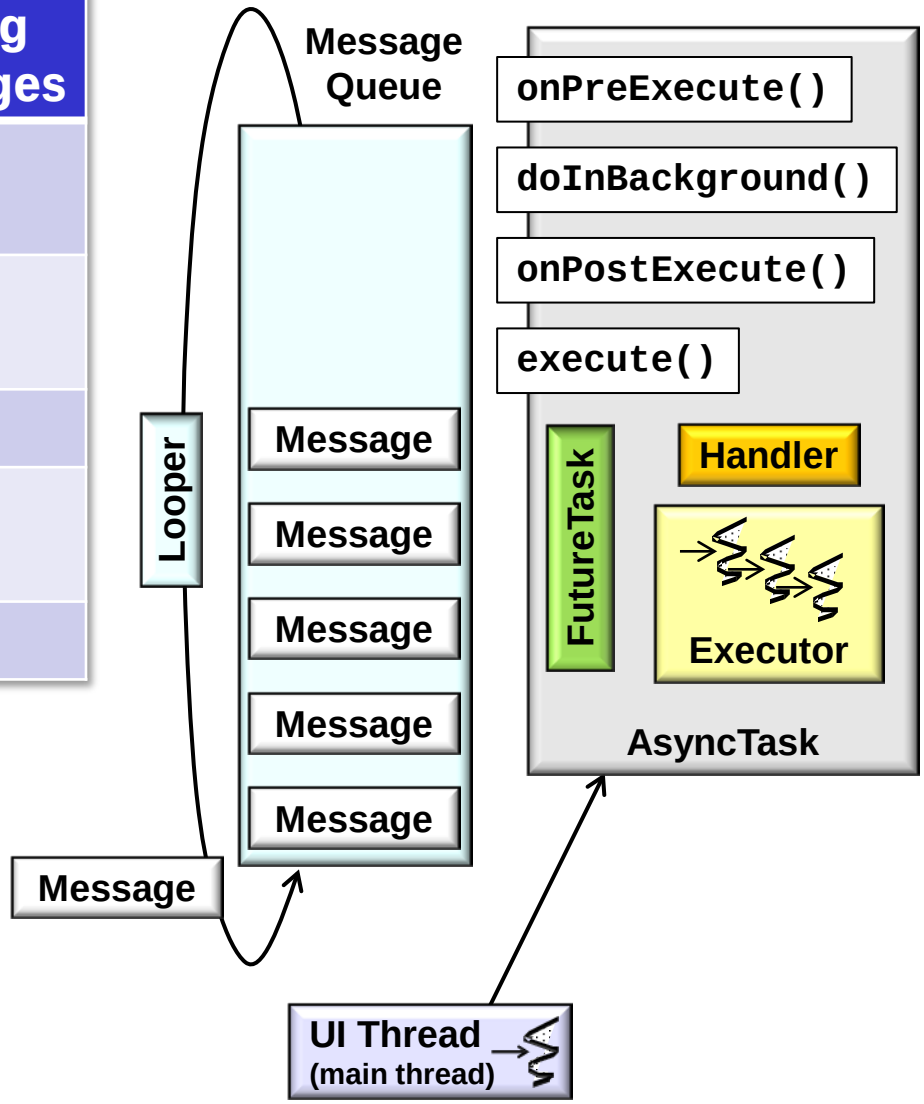
- Enables relatively transparent scalability via its Thread Pool Executor



Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	□□□	□□□	□□
Usability (Complex)	□□□	□	□□
Scalability	□□□	□	□
Flexibility	□□	□	□□□
Efficiency	□□	□□□	□□□

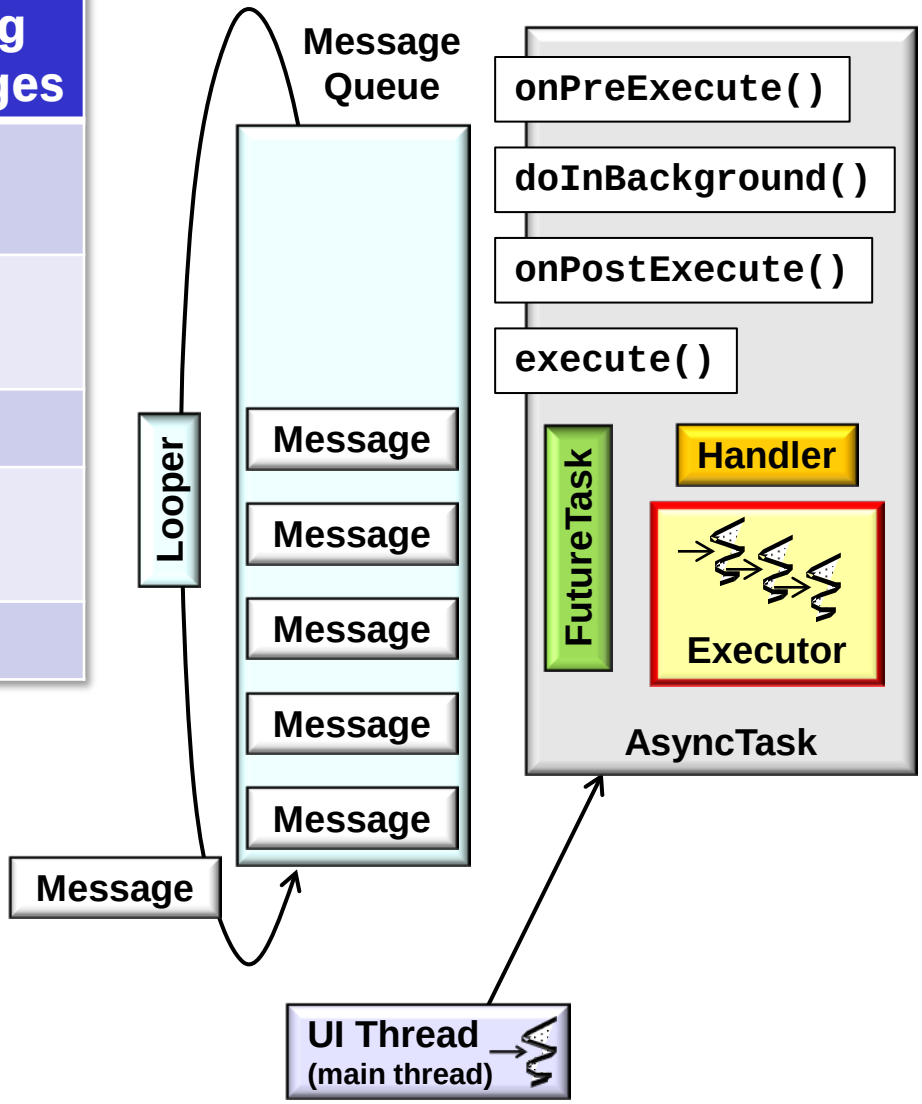
- Only intended for interactions between UI Thread & background Threads
  - But not interactions between background Threads



Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	☐☐☐	☐☐☐	☐☐
Usability (Complex)	☐☐☐	☐	☐☐
Scalability	☐☐☐	☐	☐
Flexibility	☐☐	☐	☐☐☐
Efficiency	☐☐	☐☐☐	☐☐☐

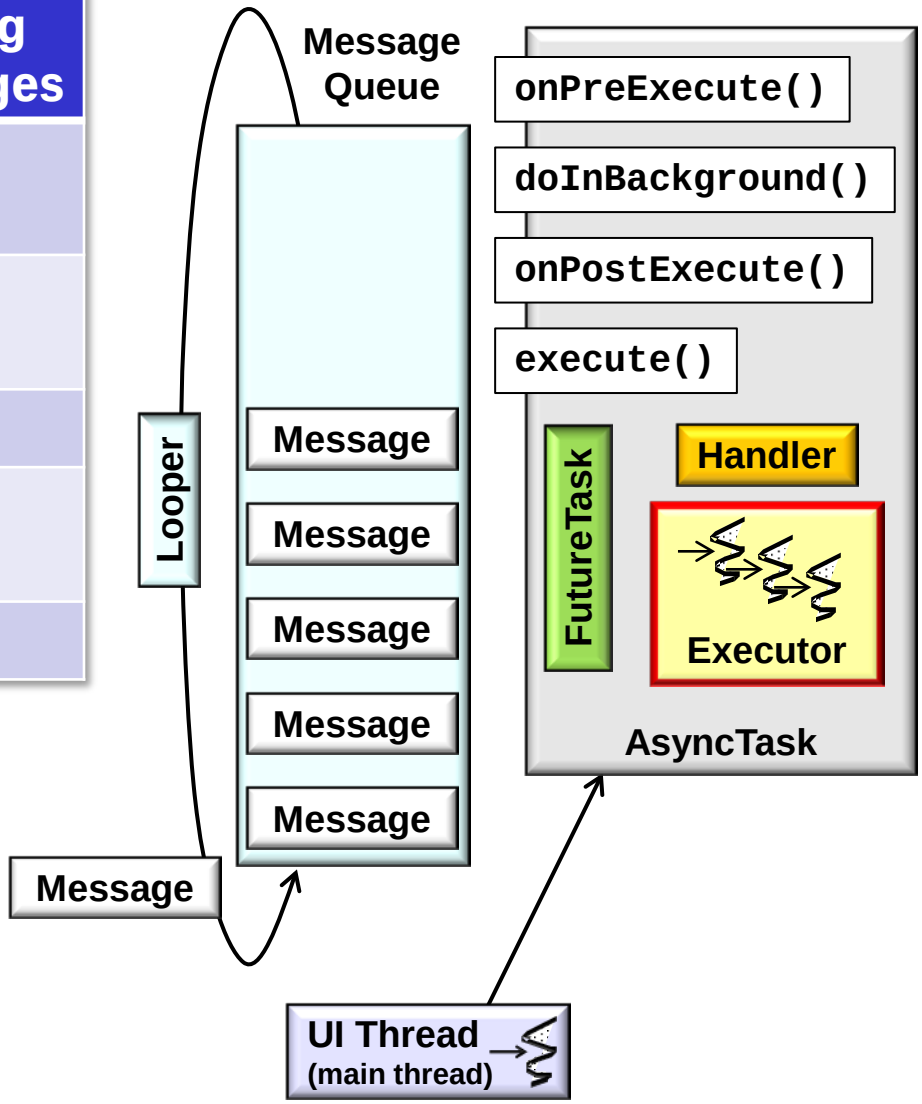
- Incurs higher overhead due to



Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	☐☐☐	☐☐☐	☐☐
Usability (Complex)	☐☐☐	☐	☐☐
Scalability	☐☐☐	☐	☐
Flexibility	☐☐	☐	☐☐☐
Efficiency	☐☐	☐☐☐	☐☐☐

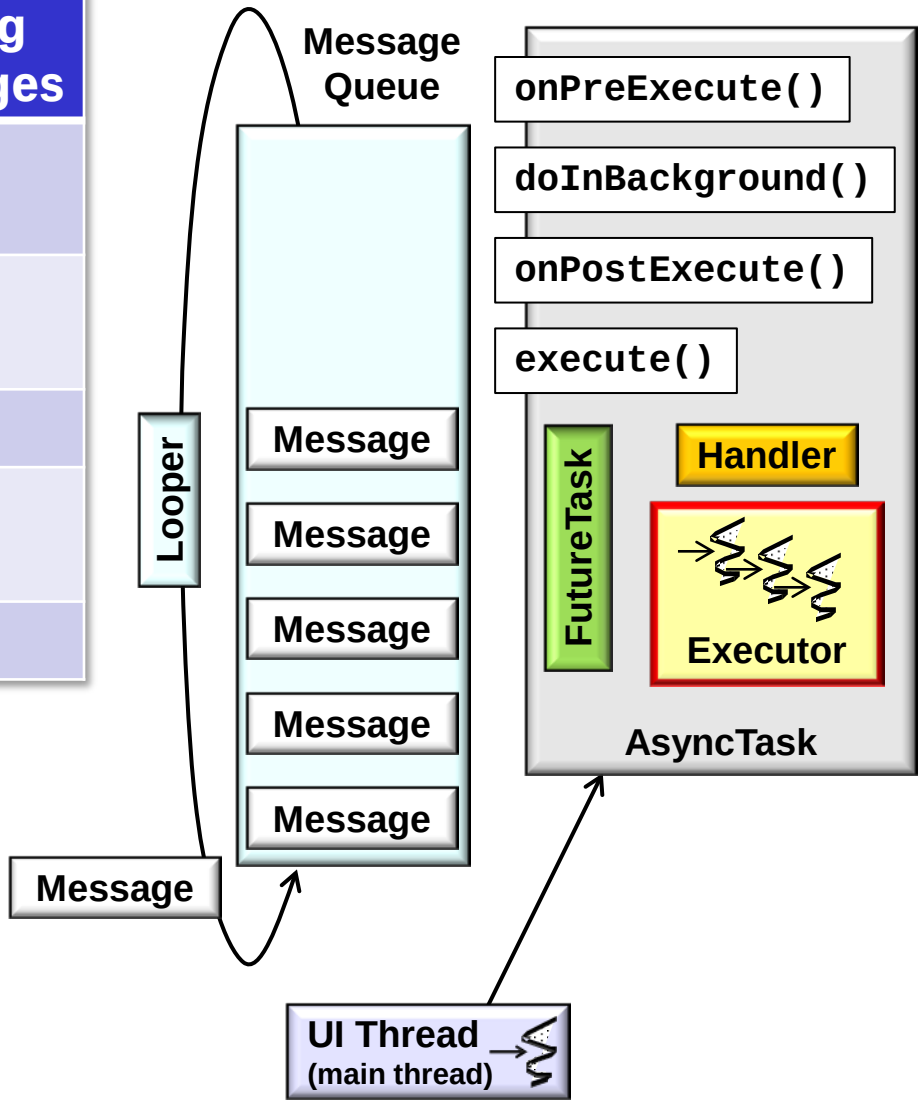
- Incurs higher overhead due to
  - Extra levels of indirection



Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	□□□	□□□	□□
Usability (Complex)	□□□	□	□□
Scalability	□□□	□	□
Flexibility	□□	□	□□□
Efficiency	□□	□□□	□□□

- Incurs higher overhead due to
  - Extra levels of indirection
  - Inter-Thread communication costs
    - e.g., synchronization, context switching, & data movement



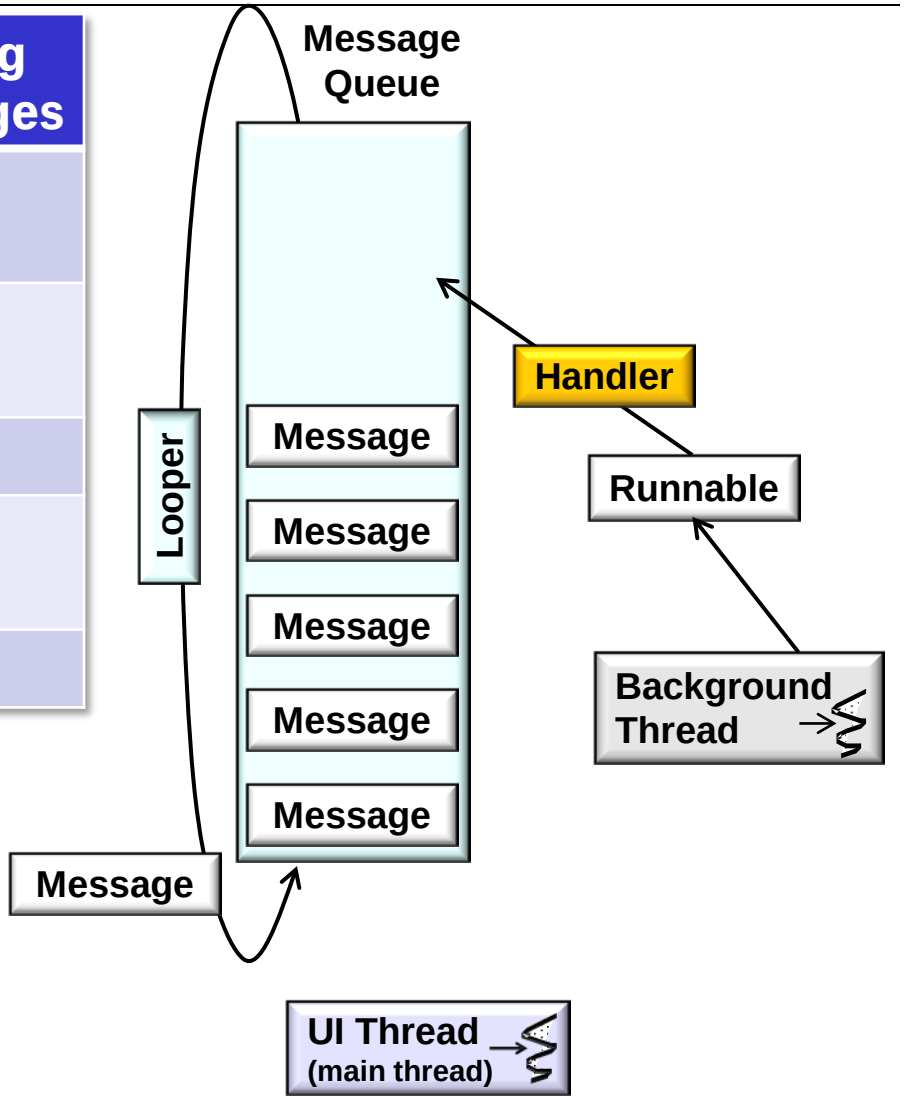
See www.dre.vanderbilt.edu/~schmidt/PDF/INFOCOM-94.pdf

Evaluating ThreadedDownloads Solutions (Part 2)

Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	☐☐☐	☐☐☐	☐☐
Usability (Complex)	☐☐☐	☐	☐☐
Scalability	☐☐☐	☐	☐
Flexibility	☐☐	☐	☐☐☐
Efficiency	☐☐	☐☐☐	☐☐☐

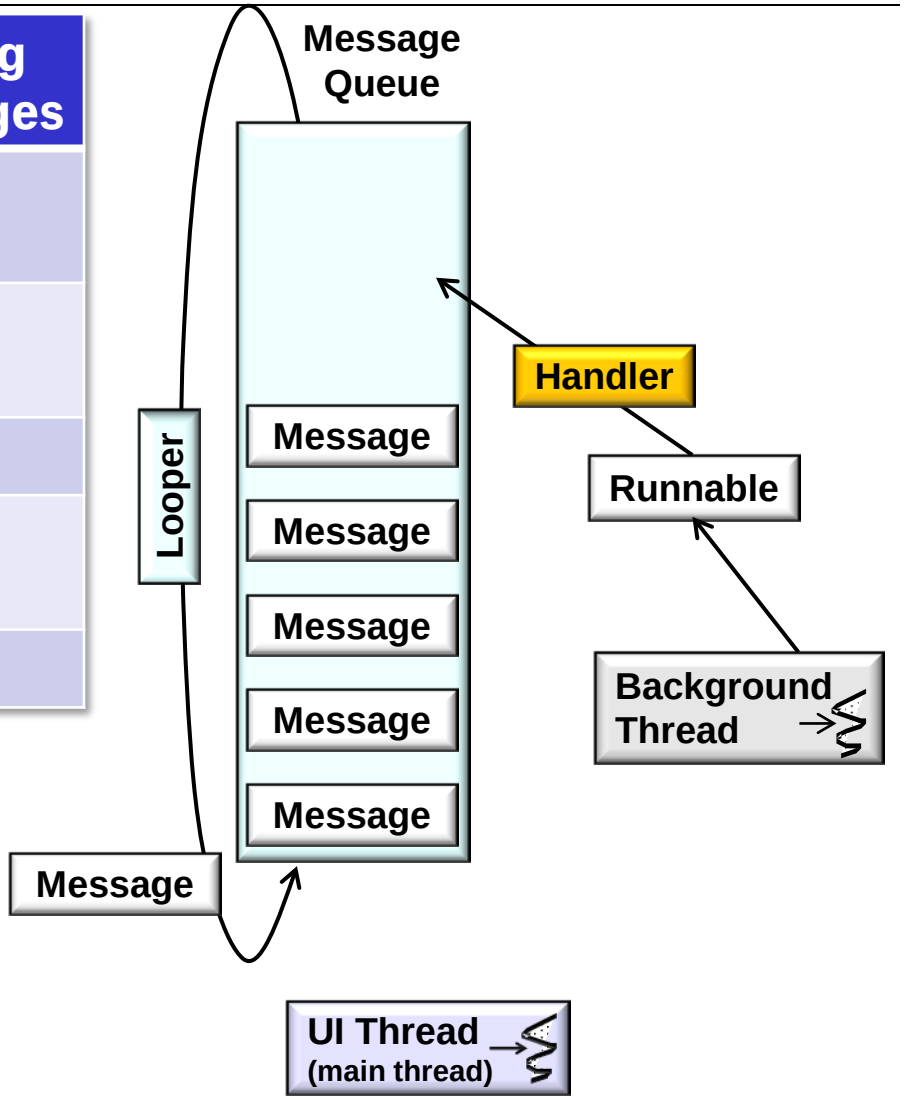
- Efficient & easy to use for simple use cases



Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	☐☐☐	☐☐☐	☐☐
Usability (Complex)	☐☐☐	☐	☐☐
Scalability	☐☐☐	☐	☐
Flexibility	☐☐	☐	☐☐☐
Efficiency	☐☐	☐☐☐	☐☐☐

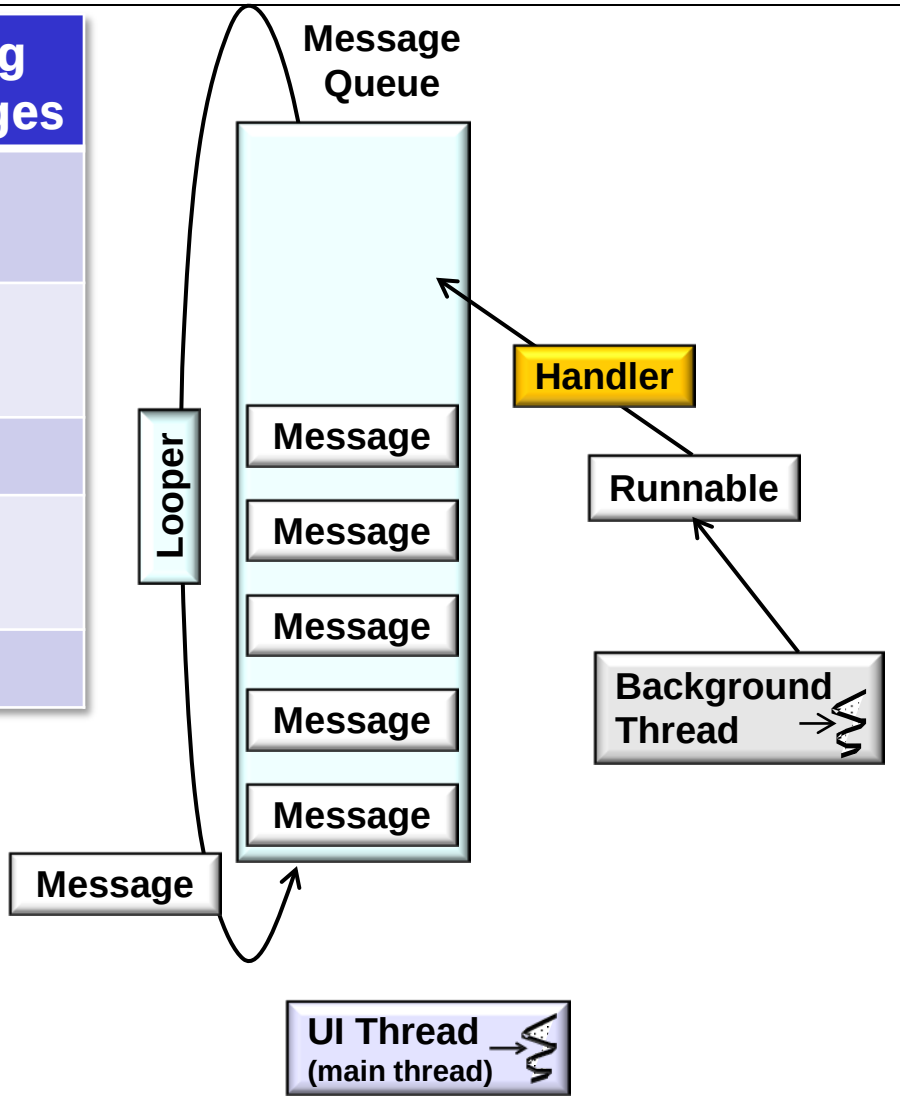
- Less flexible & easy to use for more complex use cases



Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	☐☐☐	☐☐☐	☐☐
Usability (Complex)	☐☐☐	☐	☐☐
Scalability	☐☐☐	☐	☐
Flexibility	☐☐	☐	☐☐☐
Efficiency	☐☐	☐☐☐	☐☐☐

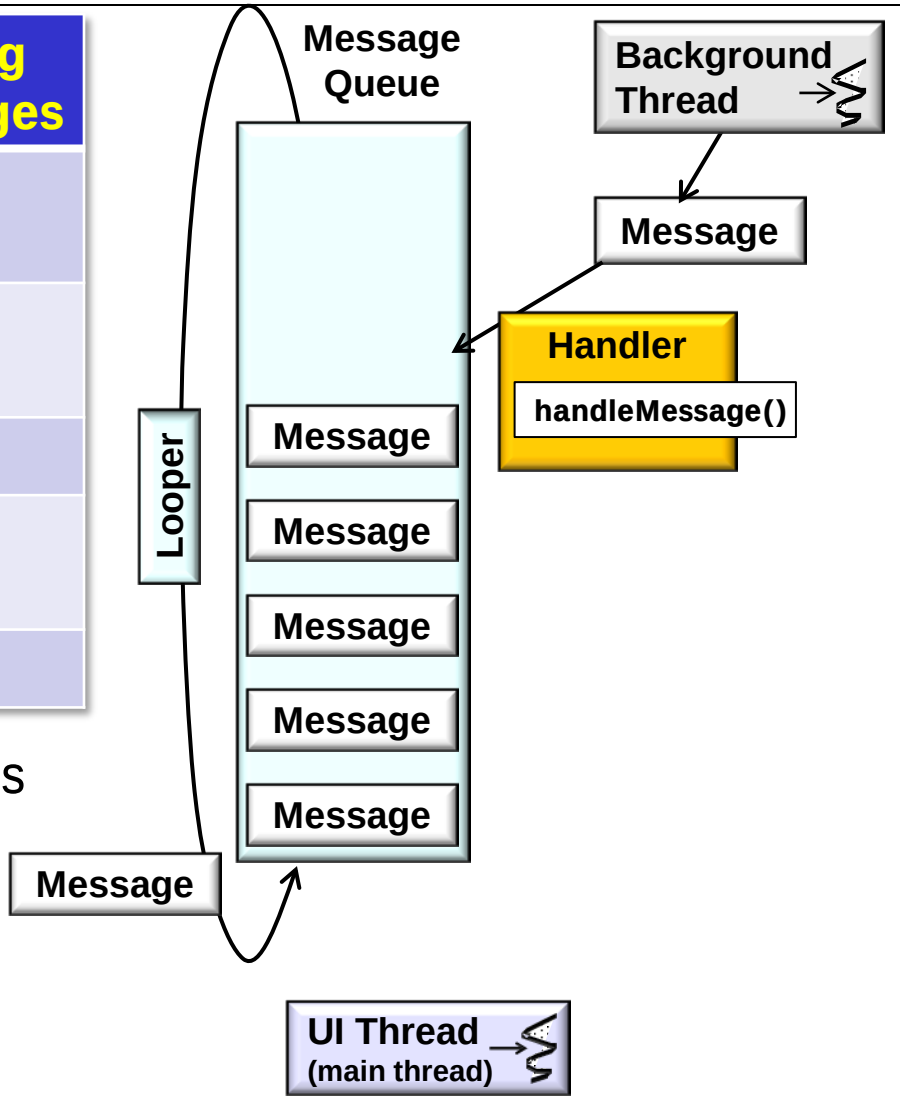
- Thread pools must be managed manually, which is hard to scale



Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	☐☐☐	☐☐☐	☐☐
Usability (Complex)	☐☐☐	☐	☐☐
Scalability	☐☐☐	☐	☐
Flexibility	☐☐	☐	☐☐☐
Efficiency	☐☐	☐☐☐	☐☐☐

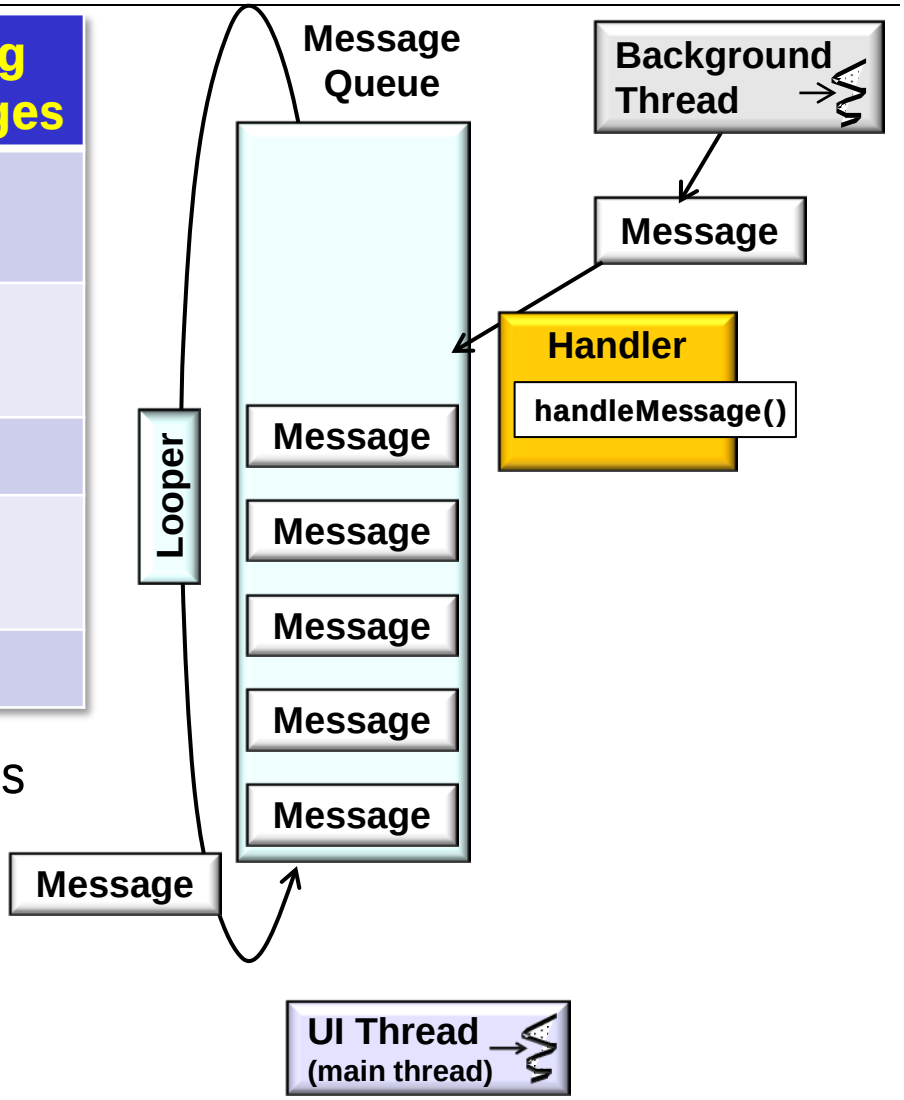
- Flexible & efficient for passing Messages between various types of Threads



Evaluating ThreadedDownloads Solutions

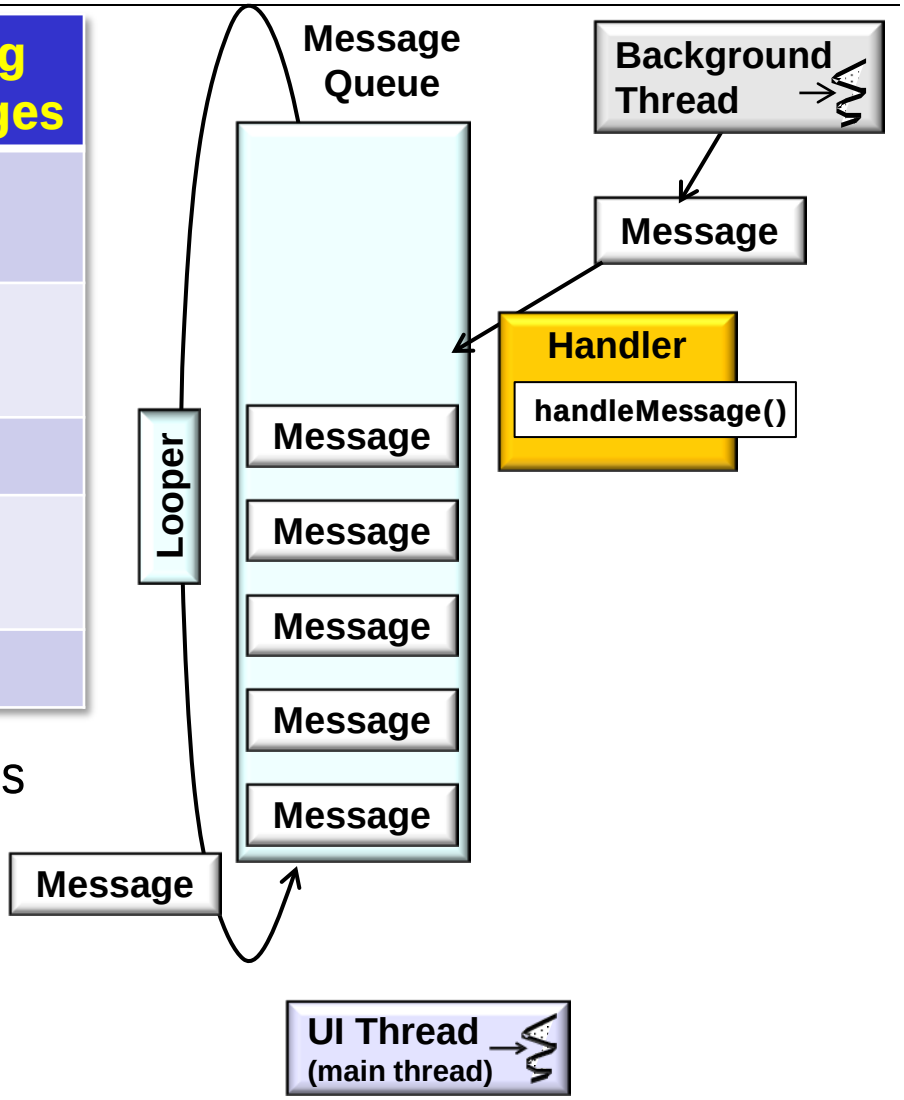
	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	☐☐☐	☐☐☐	☐☐
Usability (Complex)	☐☐☐	☐	☐☐
Scalability	☐☐☐	☐	☐
Flexibility	☐☐	☐	☐☐☐
Efficiency	☐☐	☐☐☐	☐☐☐

- Flexible & efficient for passing Messages between various types of Threads
  - Messages contain arbitrary data



Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	☐☐☐	☐☐☐	☐☐
Usability (Complex)	☐☐☐	☐	☐☐
Scalability	☐☐☐	☐	☐
Flexibility	☐☐	☐	☐☐☐
Efficiency	☐☐	☐☐☐	☐☐☐

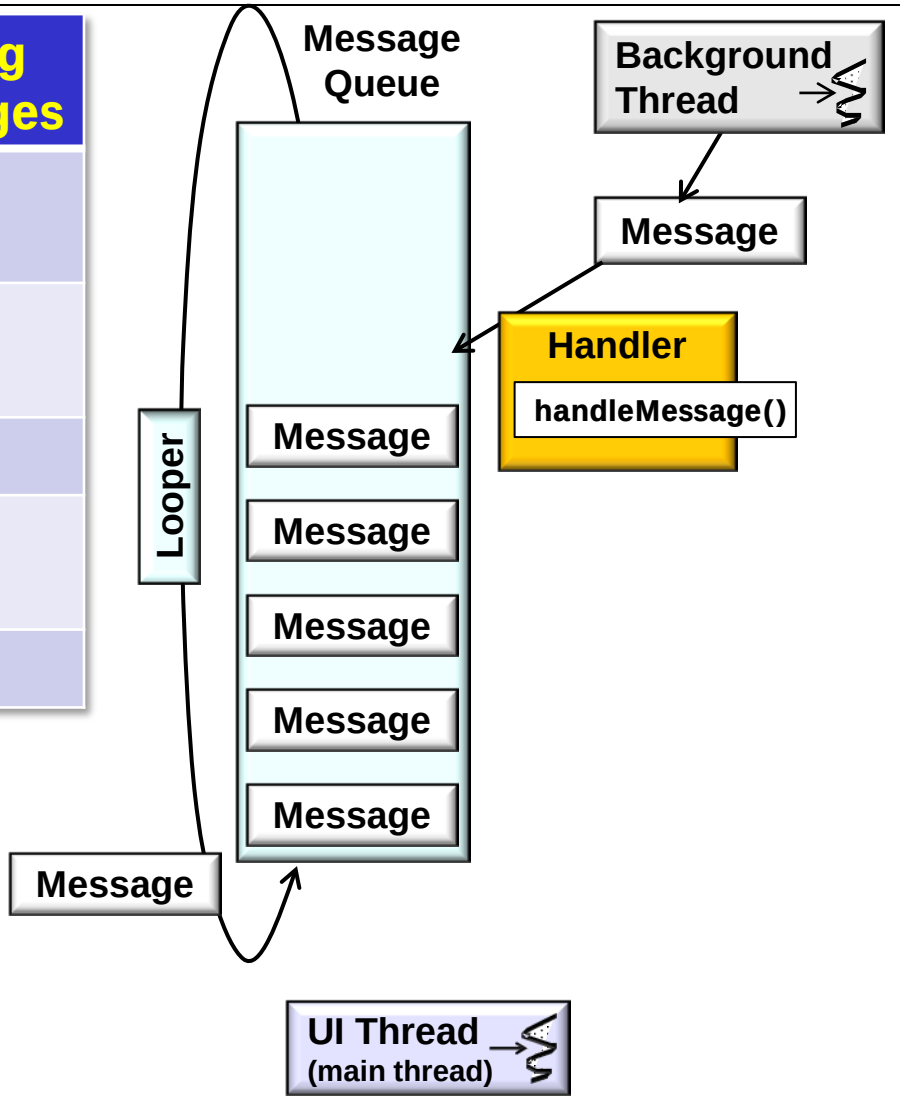


- Flexible & efficient for passing Messages between various types of Threads
  - Messages contain arbitrary data
  - Peer-to-peer conversations between various types of Threads

Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	☐☐☐	☐☐☐	☐☐
Usability (Complex)	☐☐☐	☐	☐☐
Scalability	☐☐☐	☐	☐
Flexibility	☐☐	☐	☐☐☐
Efficiency	☐☐	☐☐☐	☐☐☐

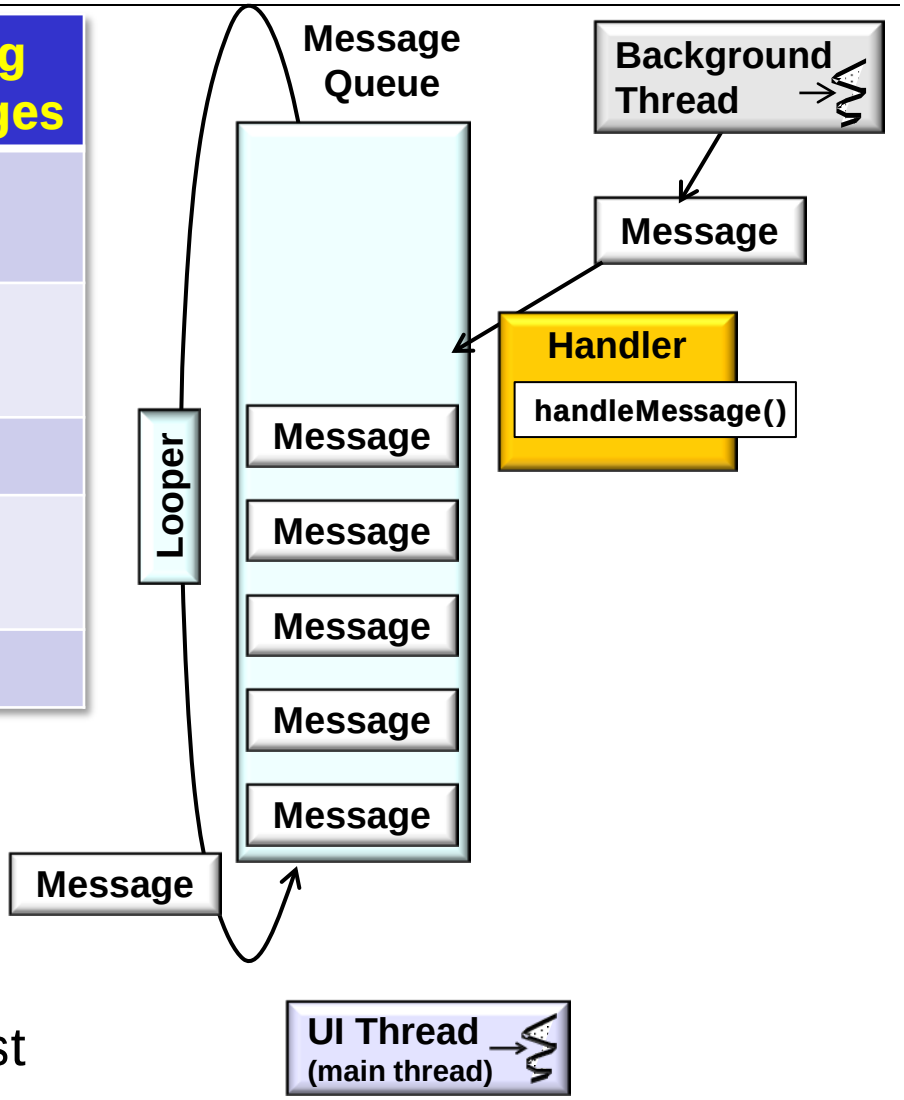
- Sending Messages is more complicated than posting Runnable commands



Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	☐☐☐	☐☐☐	☐☐
Usability (Complex)	☐☐☐	☐	☐☐
Scalability	☐☐☐	☐	☐
Flexibility	☐☐	☐	☐☐☐
Efficiency	☐☐	☐☐☐	☐☐☐

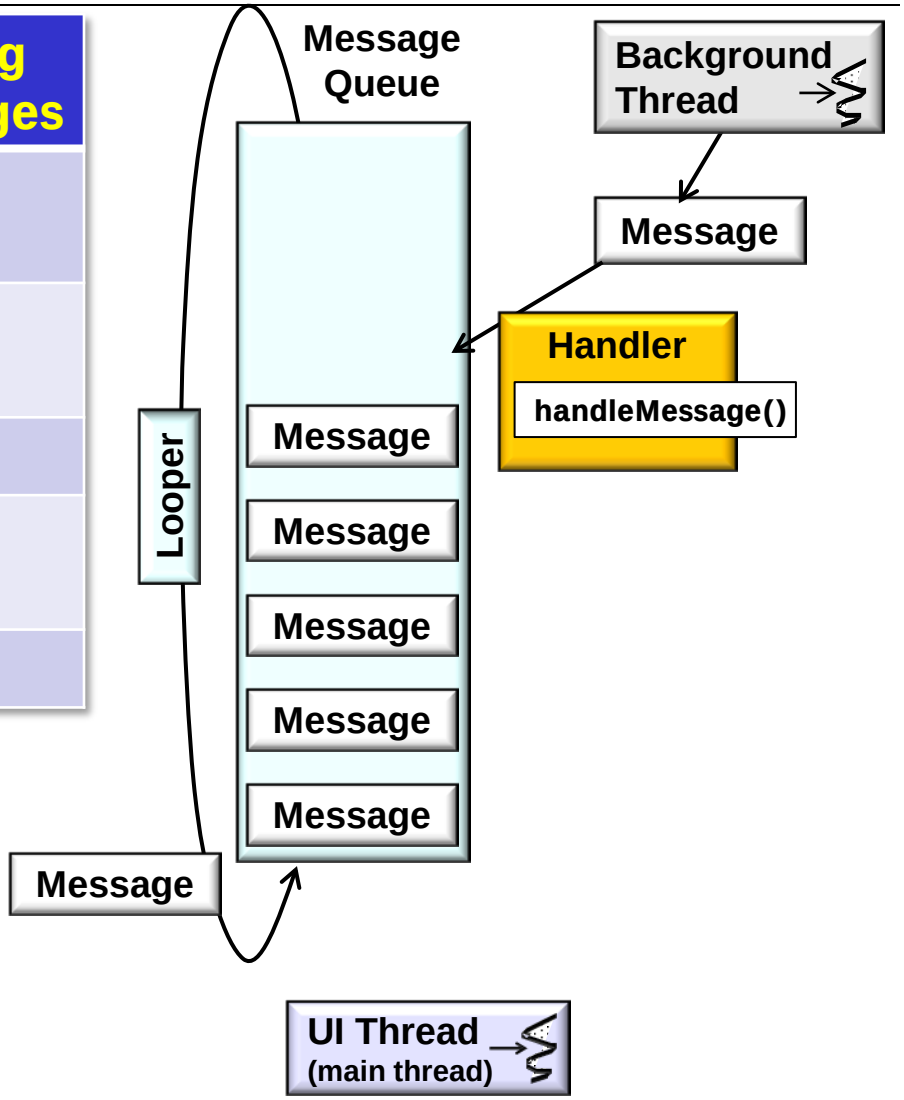
- Sending Messages is more complicated than posting Runnable commands
  - e.g., Handler must be extended & its handleMessage() hook method overridden or a Handler.Callback must be used



Evaluating ThreadedDownloads Solutions

	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	☐☐☐	☐☐☐	☐☐
Usability (Complex)	☐☐☐	☐	☐☐
Scalability	☐☐☐	☐	☐
Flexibility	☐☐	☐	☐☐☐
Efficiency	☐☐	☐☐☐	☐☐☐

- Thread pools must be managed manually, which is hard to scale



Evaluating ThreadedDownloads Solutions

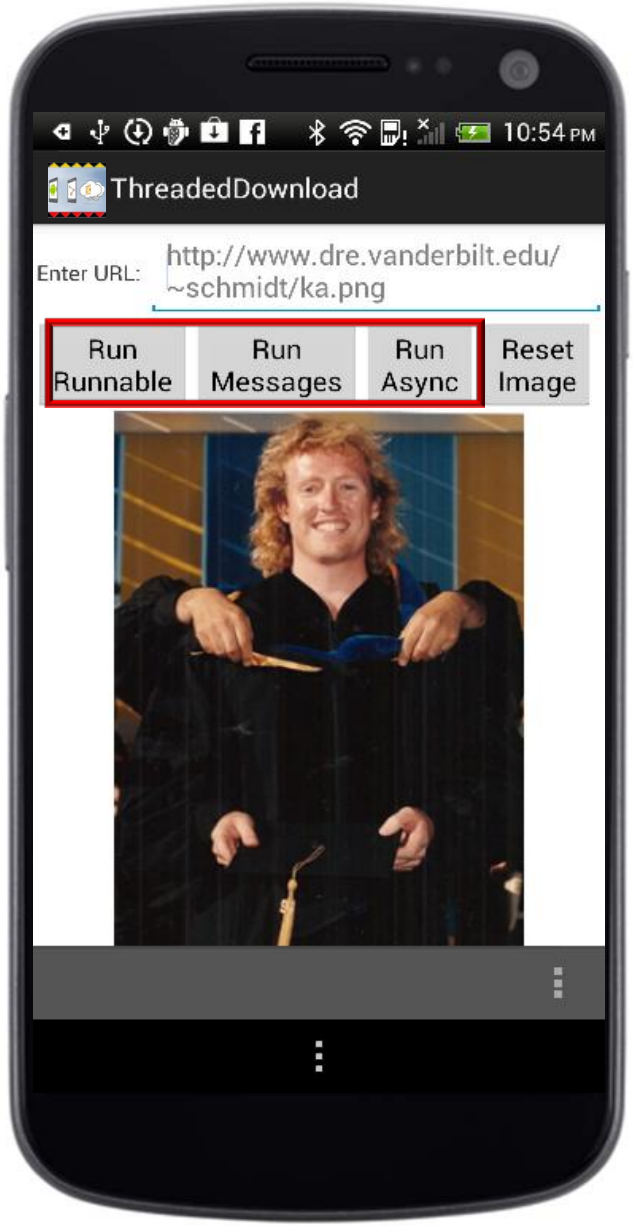
	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	☐☐☐	☐☐☐	☐☐
Usability (Complex)	☐☐☐	☐	☐☐
Scalability	☐☐☐	☐	☐
Flexibility	☐☐	☐	☐☐☐
Efficiency	☐☐	☐☐☐	☐☐☐

Choosing the right concurrency model depends on requirements & expertise

Summary

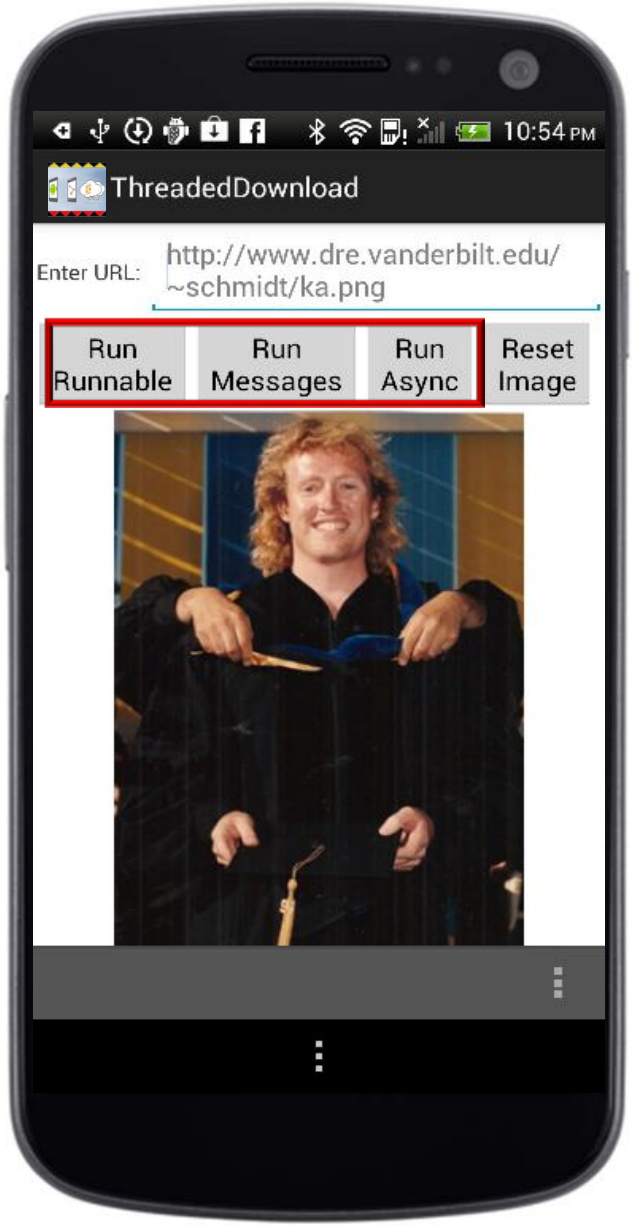
Summary

- Threaded Downloads implements three different concurrency models



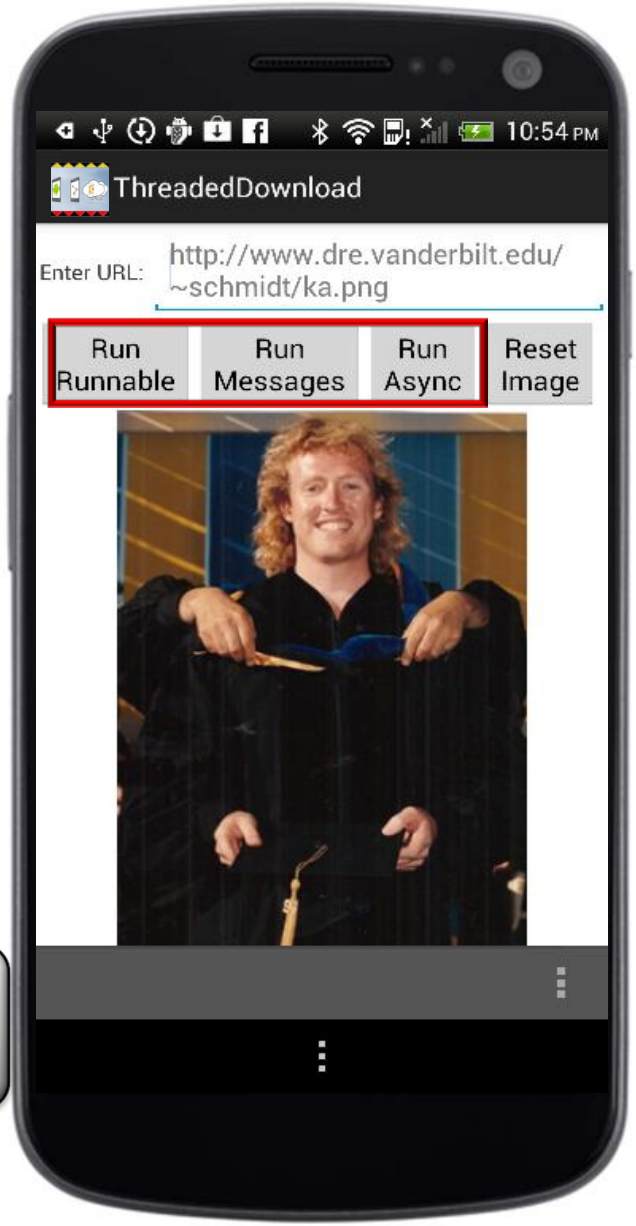
Summary

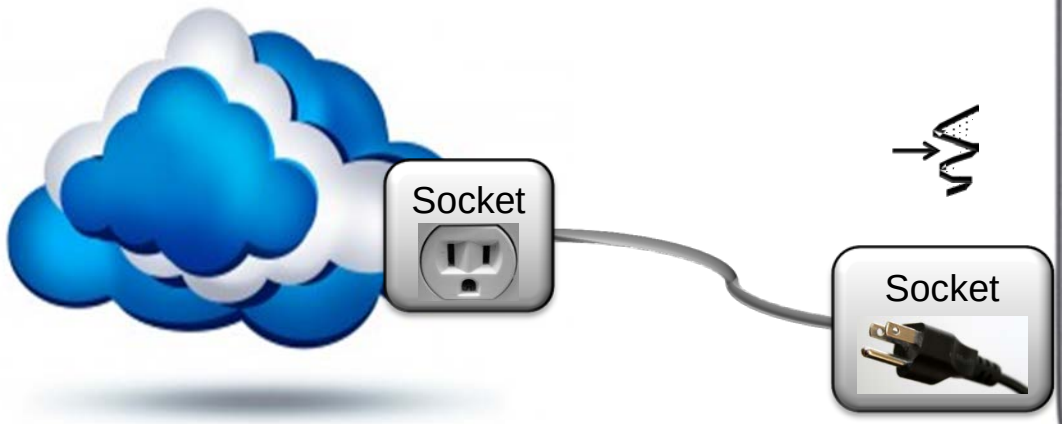
- Threaded Downloads implements three different concurrency models
  - Uses the Android HaMeR & AsyncTask frameworks



Summary

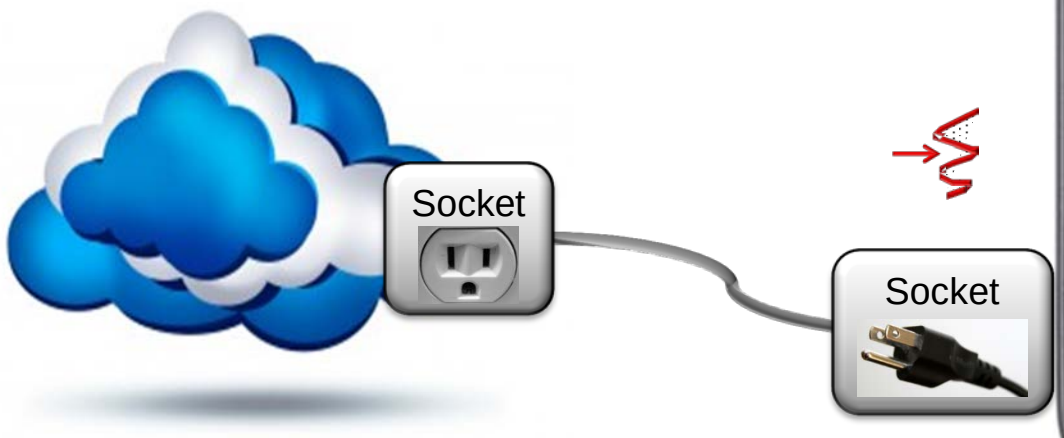
- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common

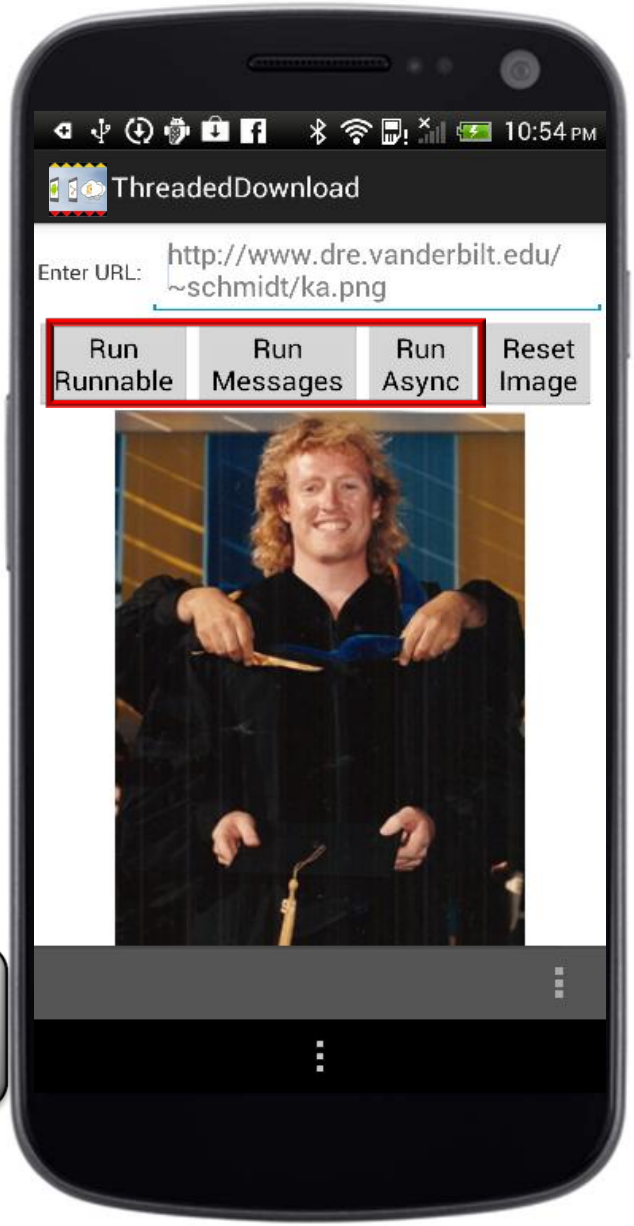




Summary

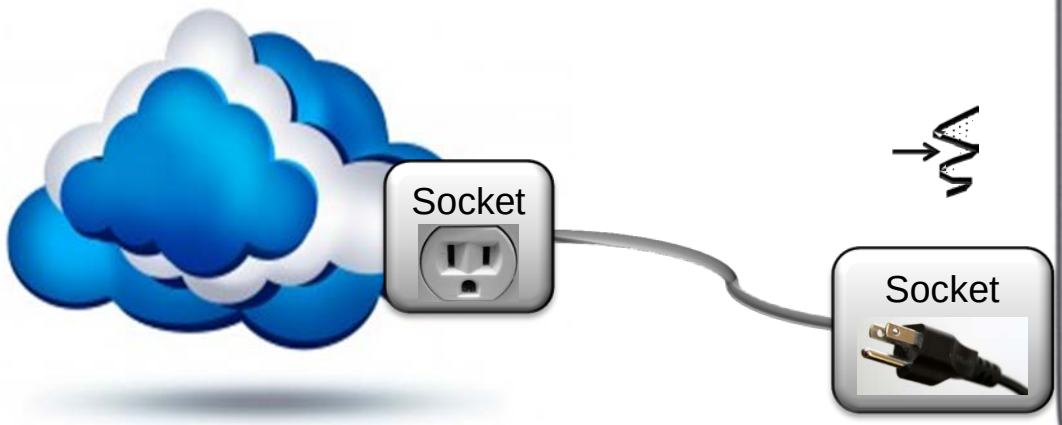
- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
  - Long duration operations run in a background Thread

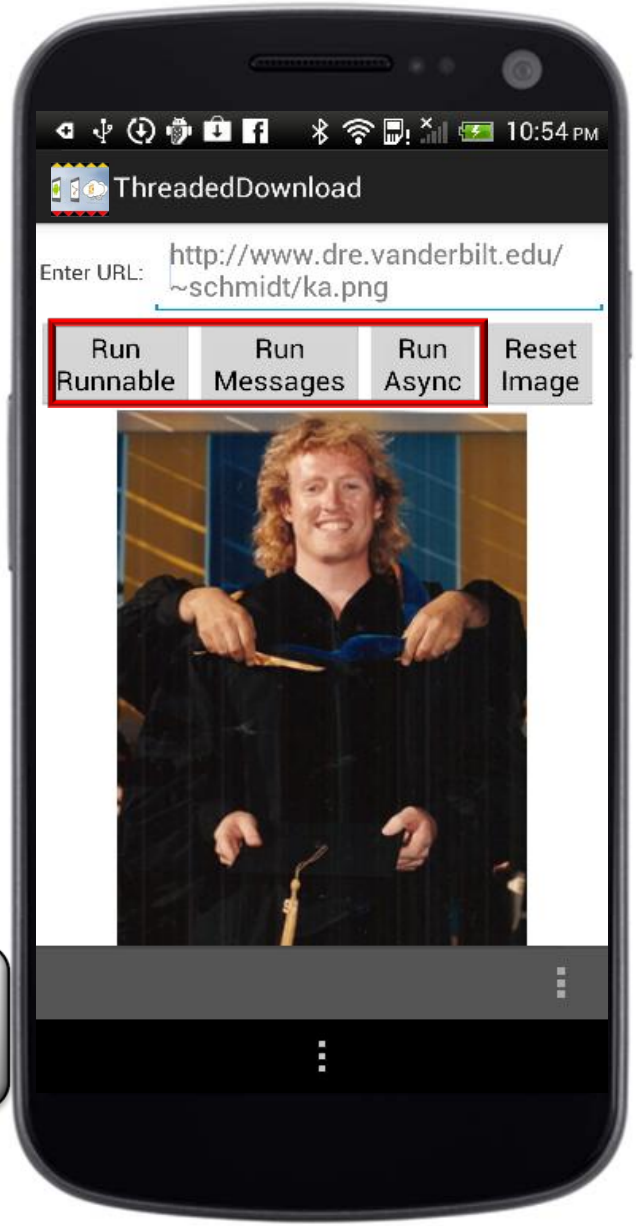




Summary

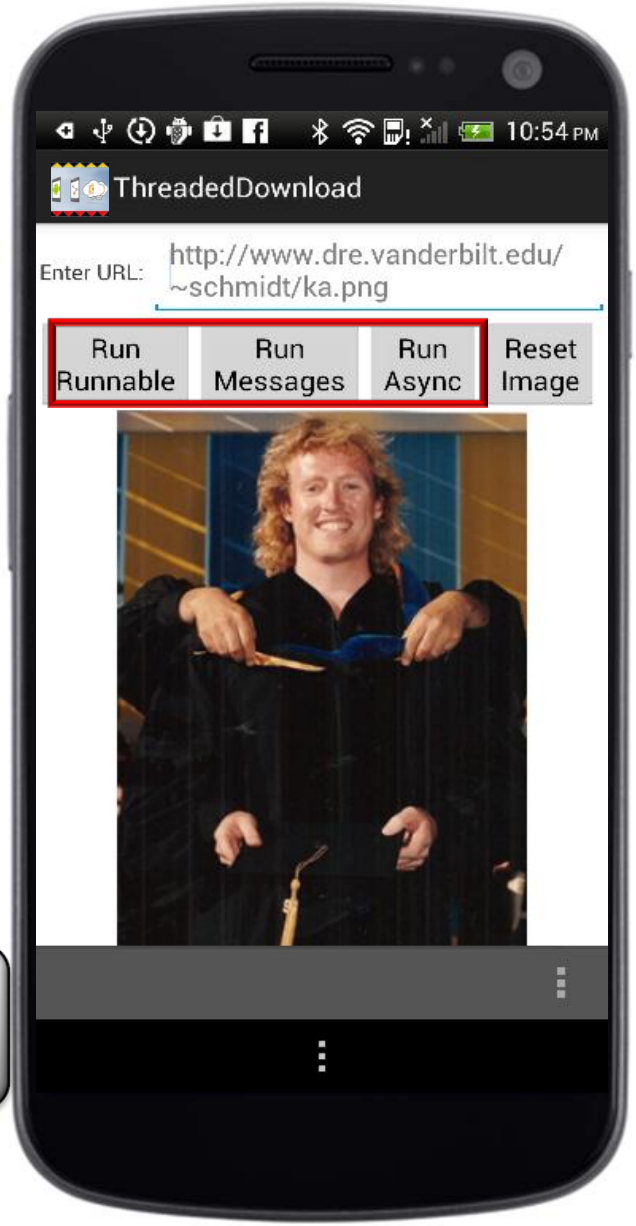
- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
  - Long duration operations run in a background Thread
  - Short duration operations run in the UI Thread

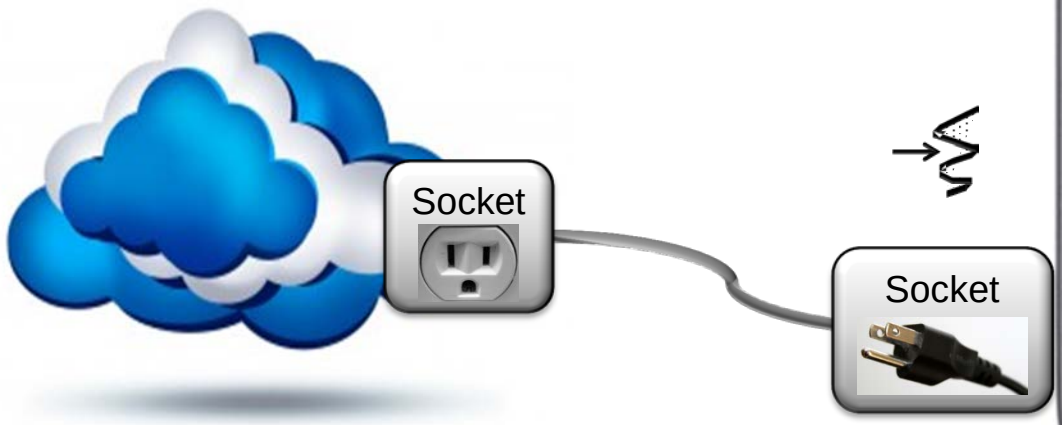




Summary

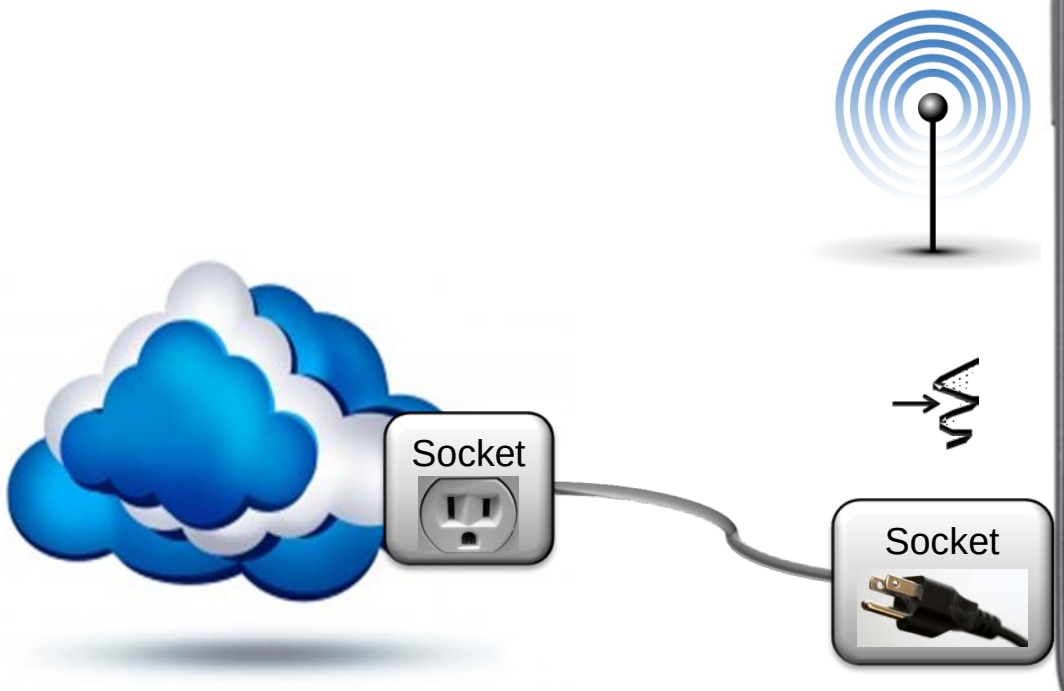
- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences

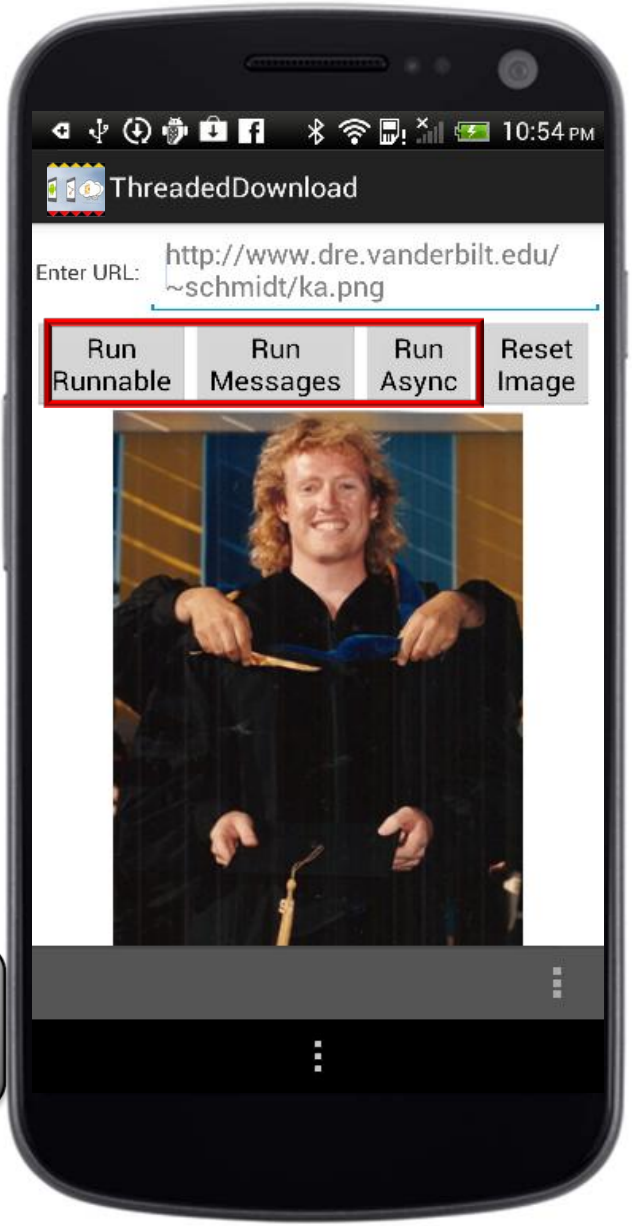




Summary

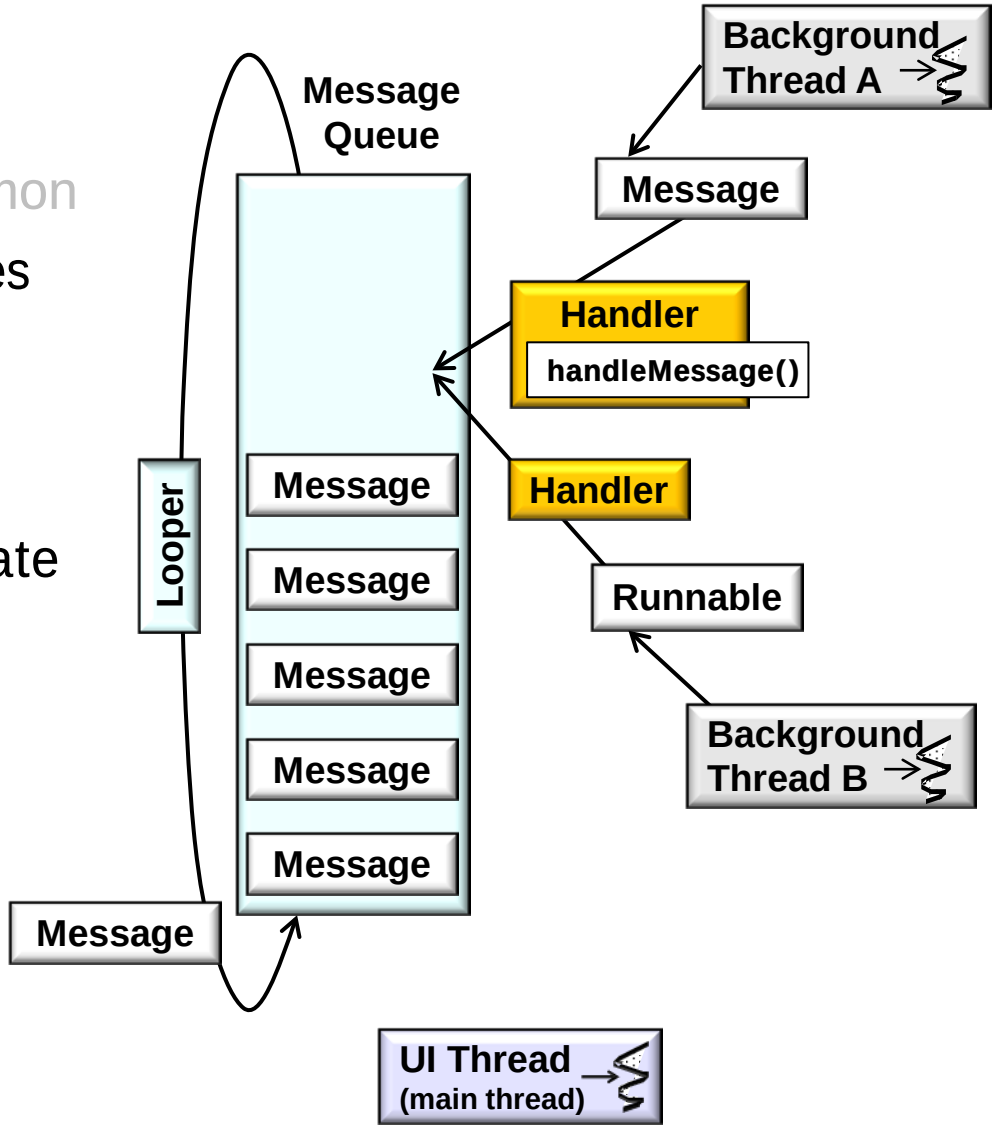
- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
  - Background Threads & UI Thread communicate in different ways





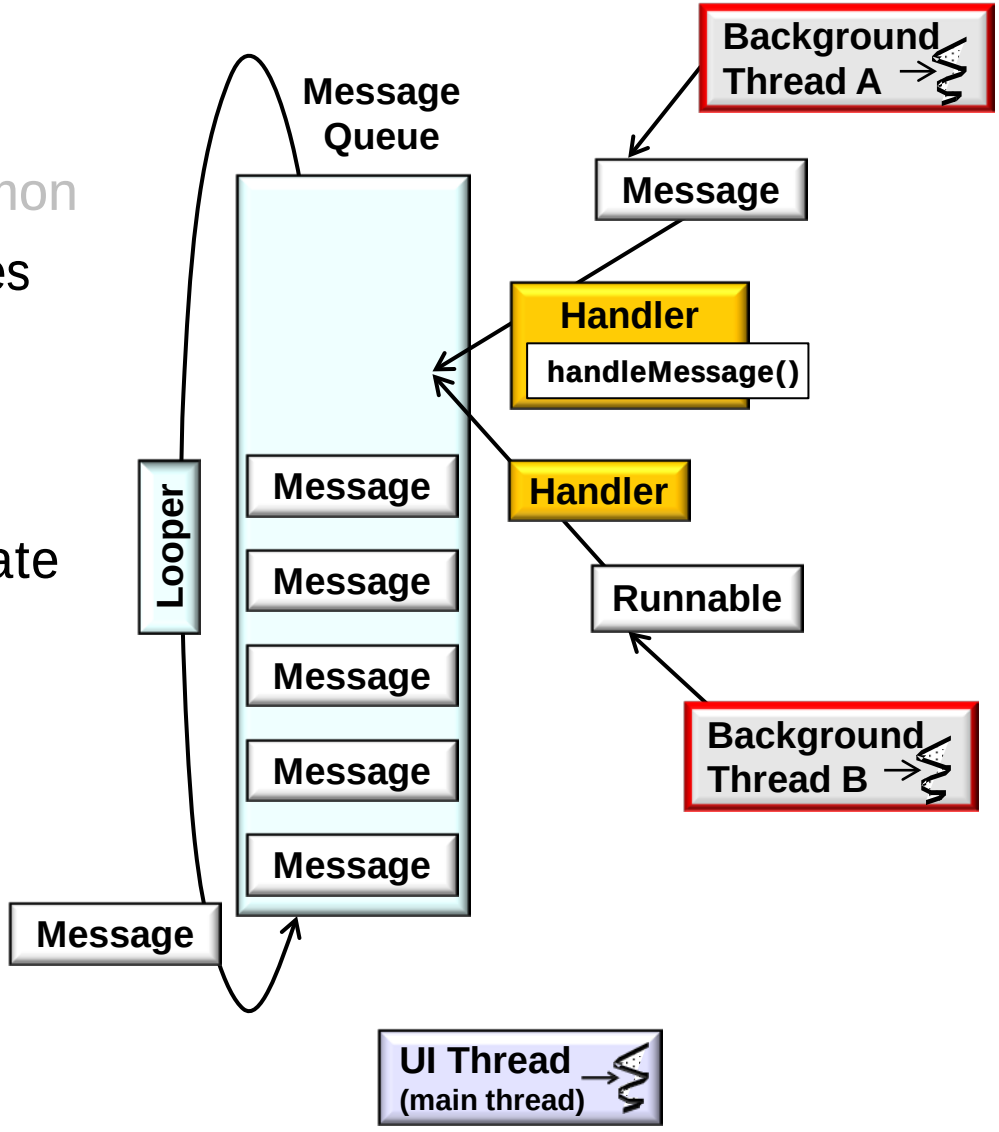
Summary

- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
  - Background Threads & UI Thread communicate in different ways
    - Background Threads in HaMeR framework explicitly communicate with UI Thread



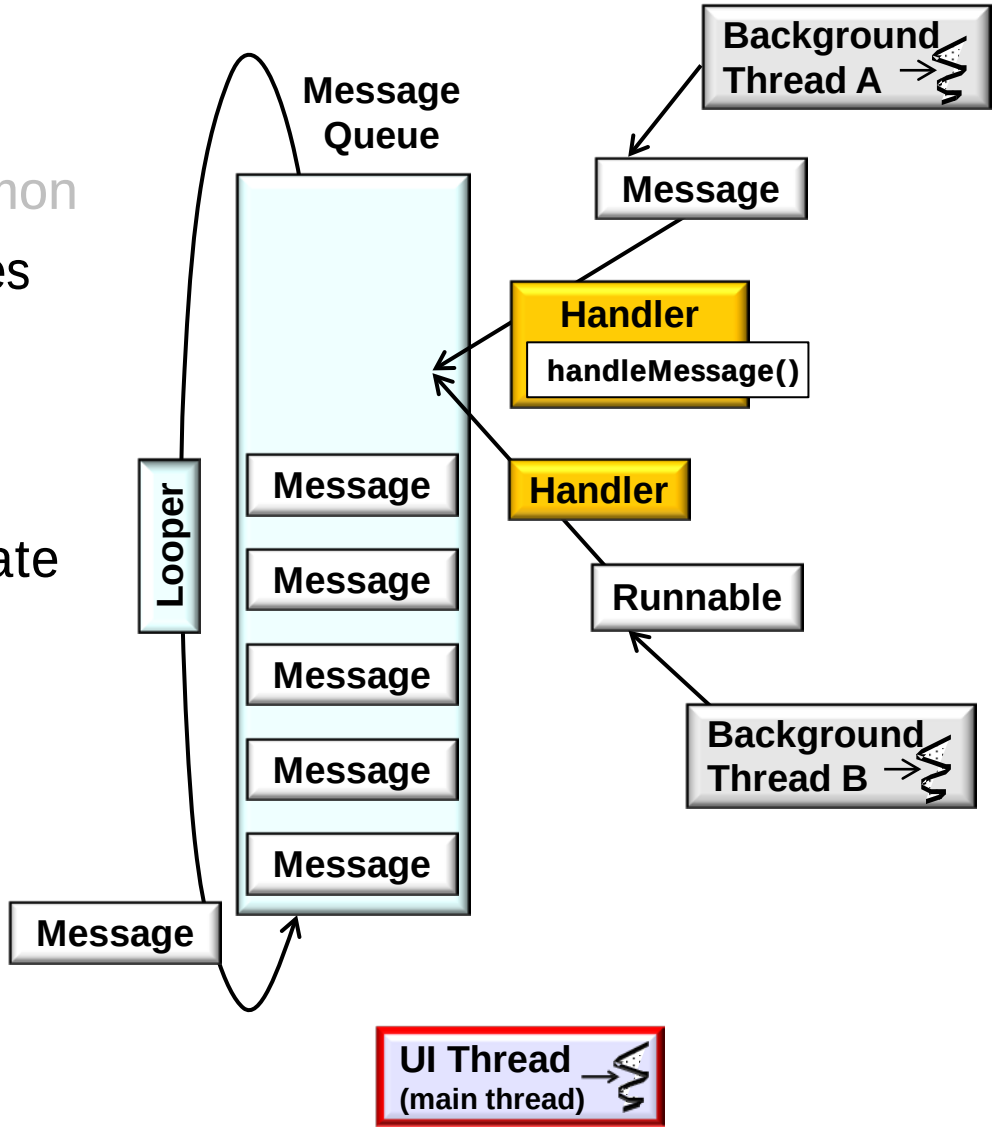
Summary

- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
  - Background Threads & UI Thread communicate in different ways
    - Background Threads in HaMeR framework explicitly communicate with UI Thread



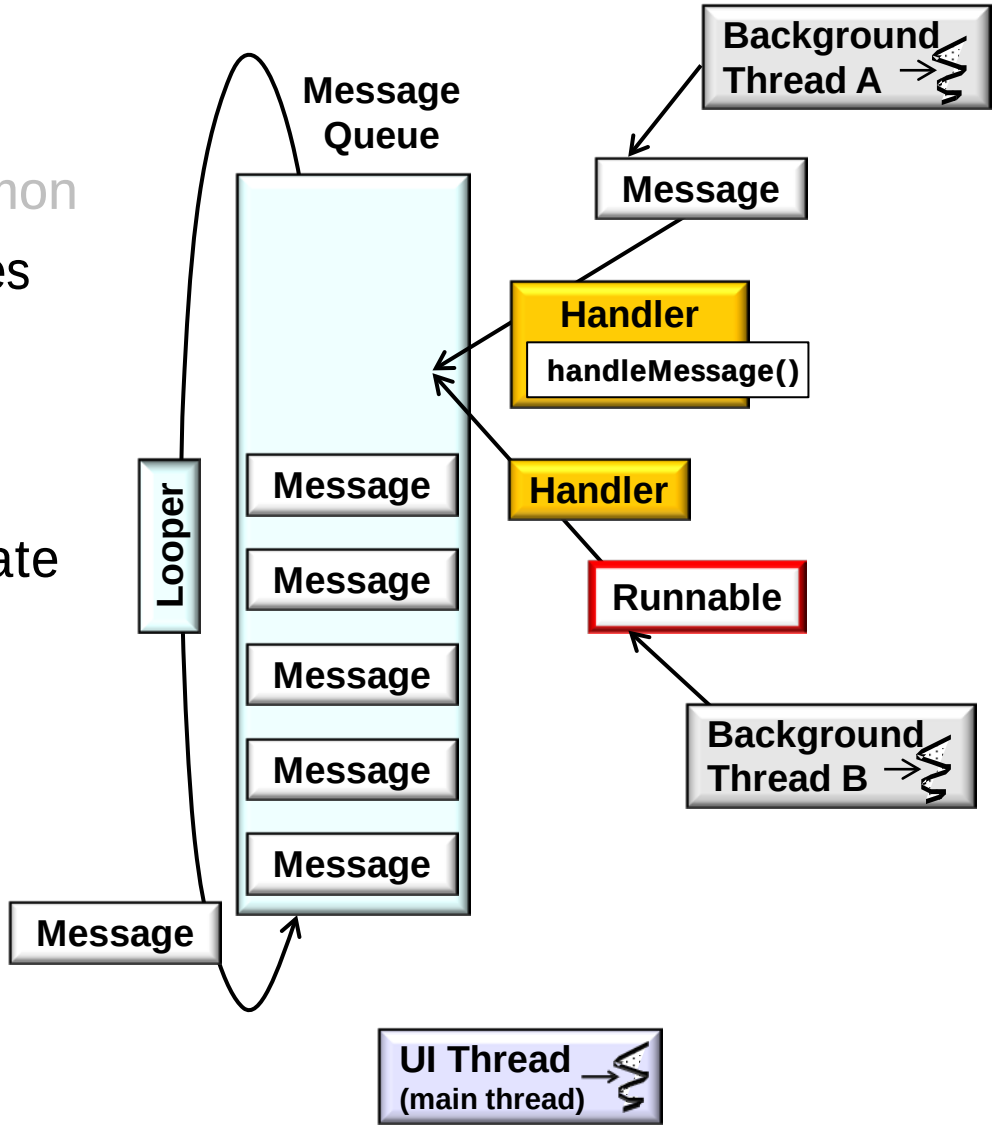
Summary

- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
  - Background Threads & UI Thread communicate in different ways
    - Background Threads in HaMeR framework explicitly communicate with UI Thread



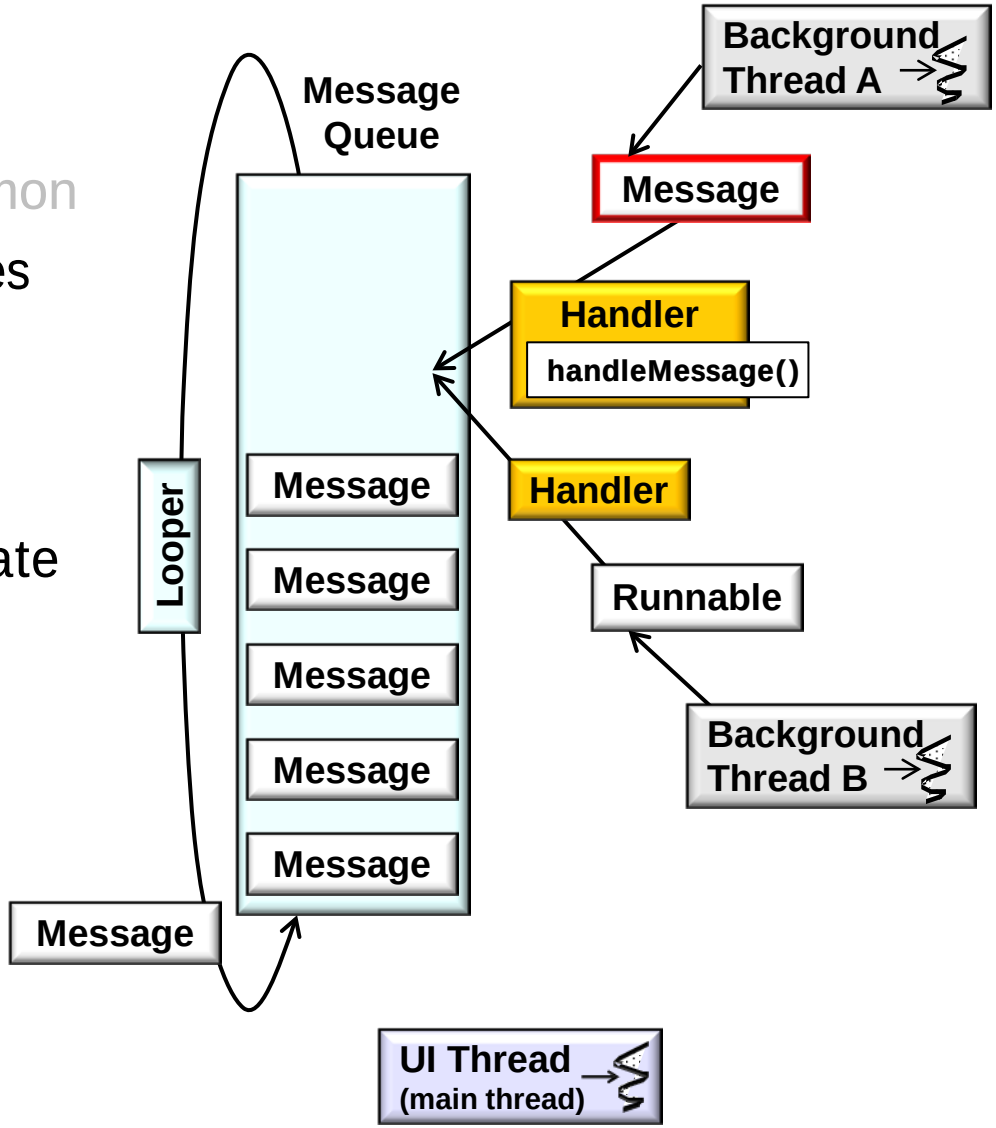
Summary

- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
  - Background Threads & UI Thread communicate in different ways
    - Background Threads in HaMeR framework explicitly communicate with UI Thread



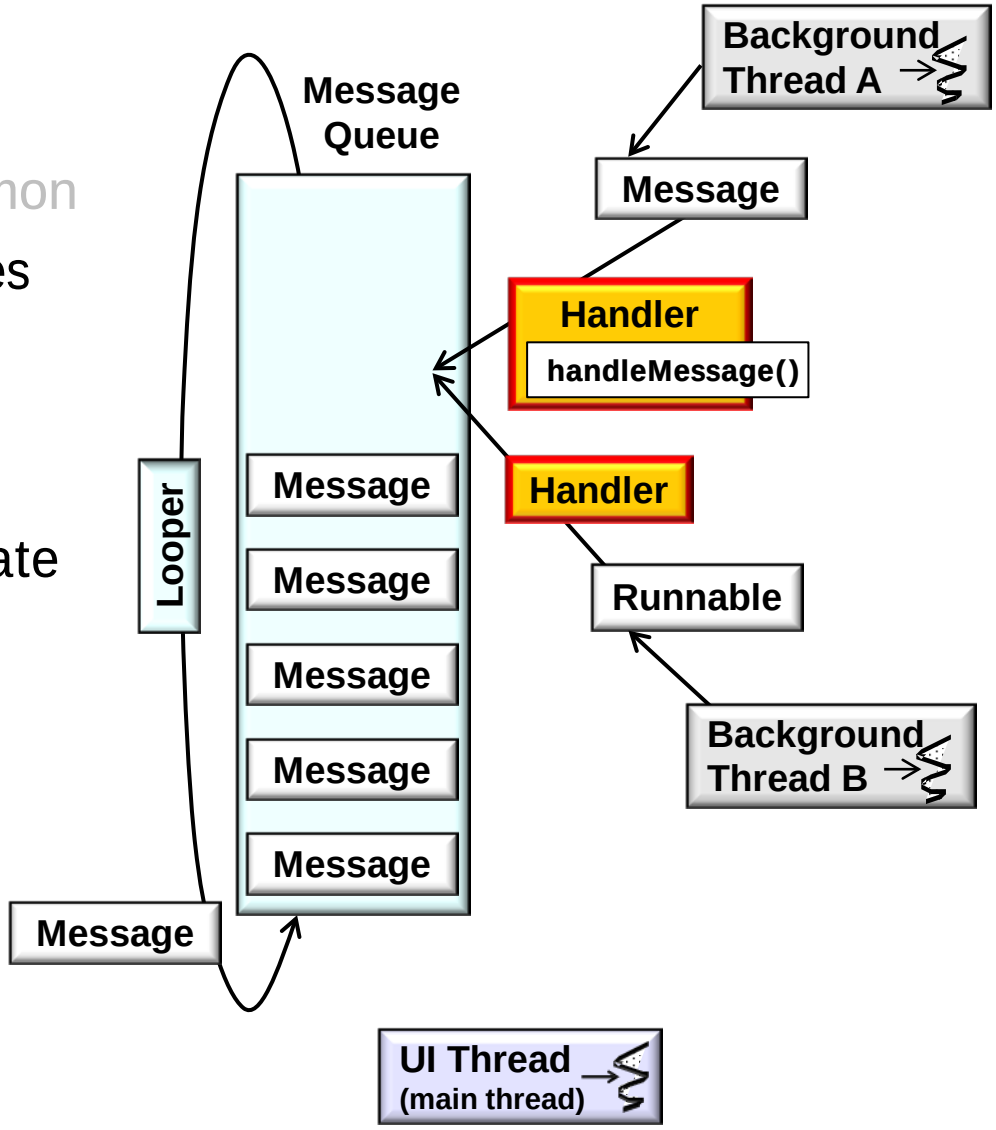
Summary

- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
  - Background Threads & UI Thread communicate in different ways
    - Background Threads in HaMeR framework explicitly communicate with UI Thread



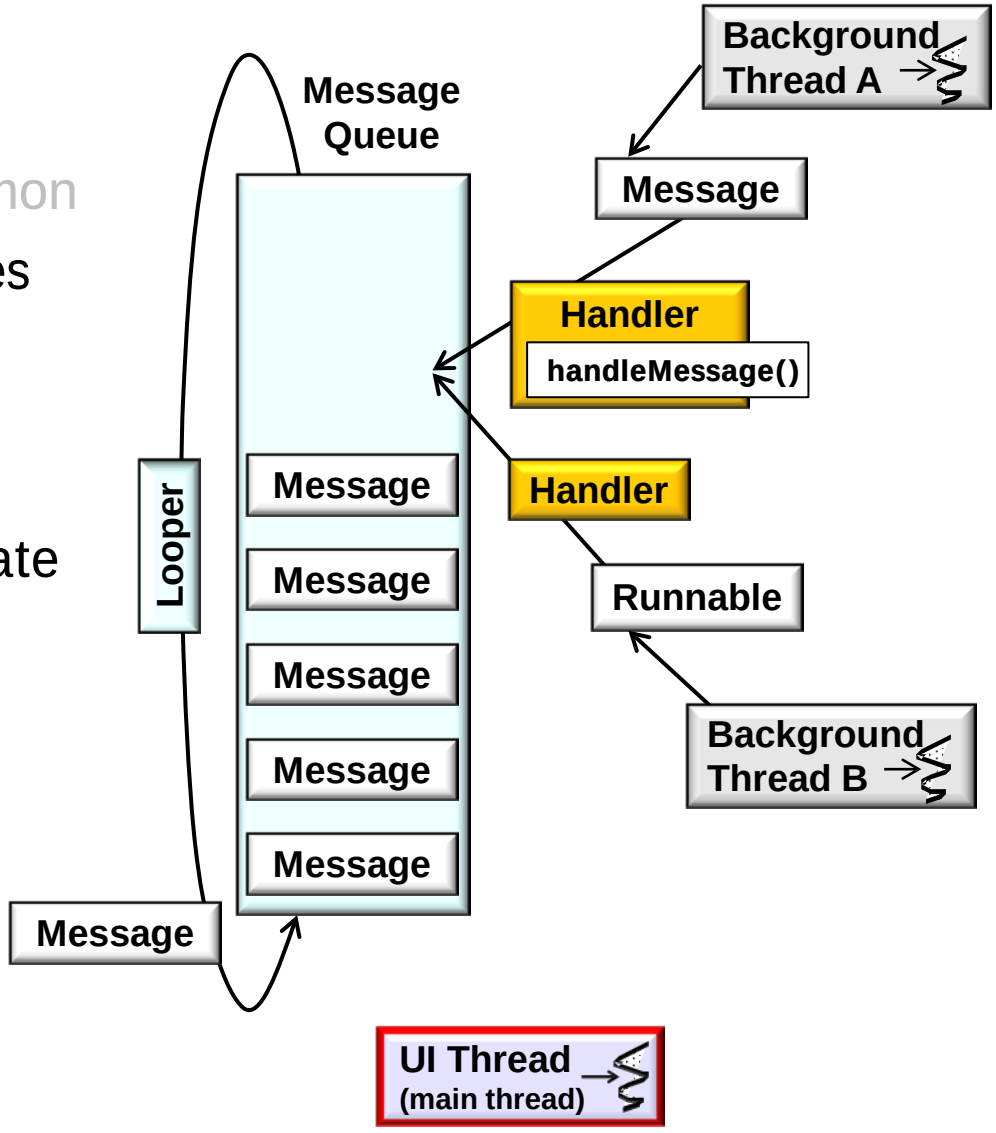
Summary

- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
  - Background Threads & UI Thread communicate in different ways
    - Background Threads in HaMeR framework explicitly communicate with UI Thread



Summary

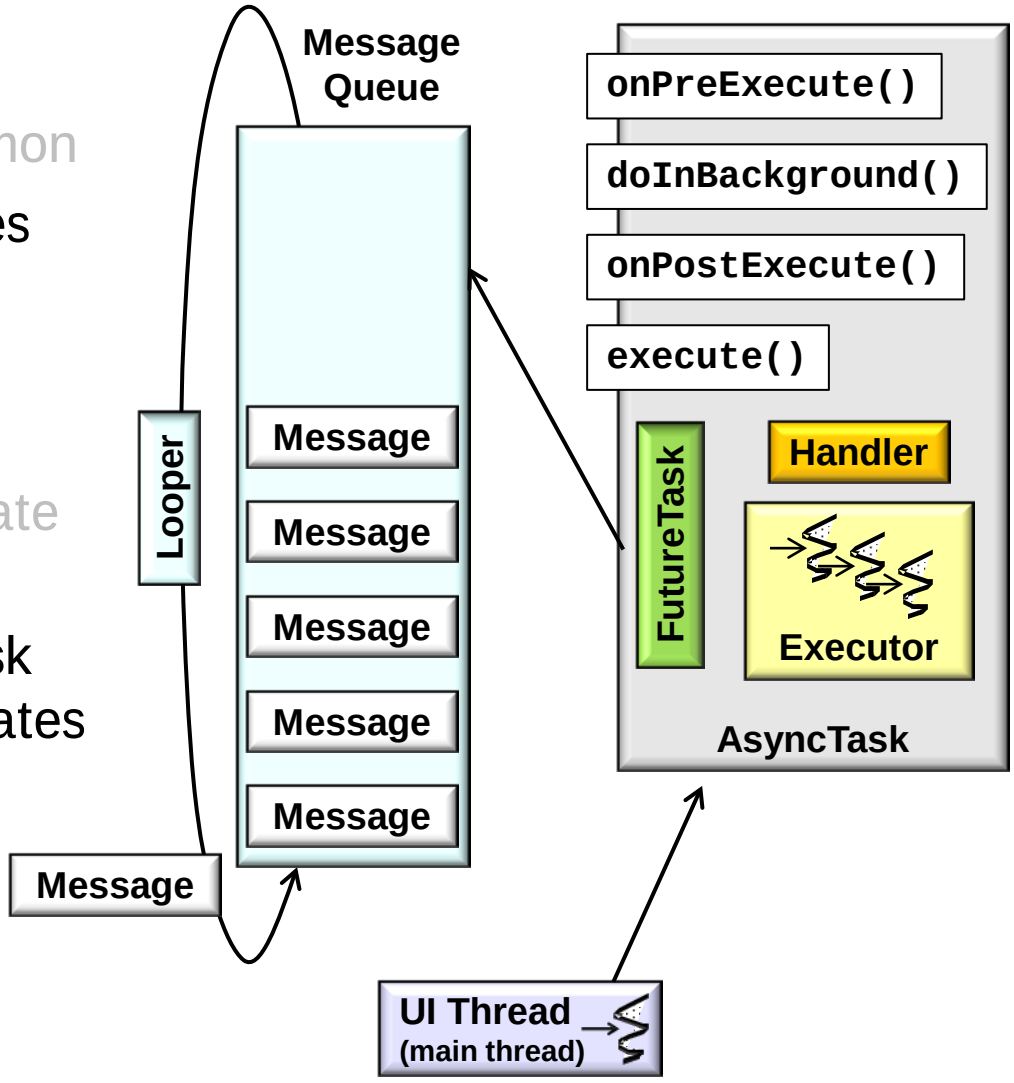
- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
  - Background Threads & UI Thread communicate in different ways
    - Background Threads in HaMeR framework explicitly communicate with UI Thread



See earlier parts on using the Android Handler class

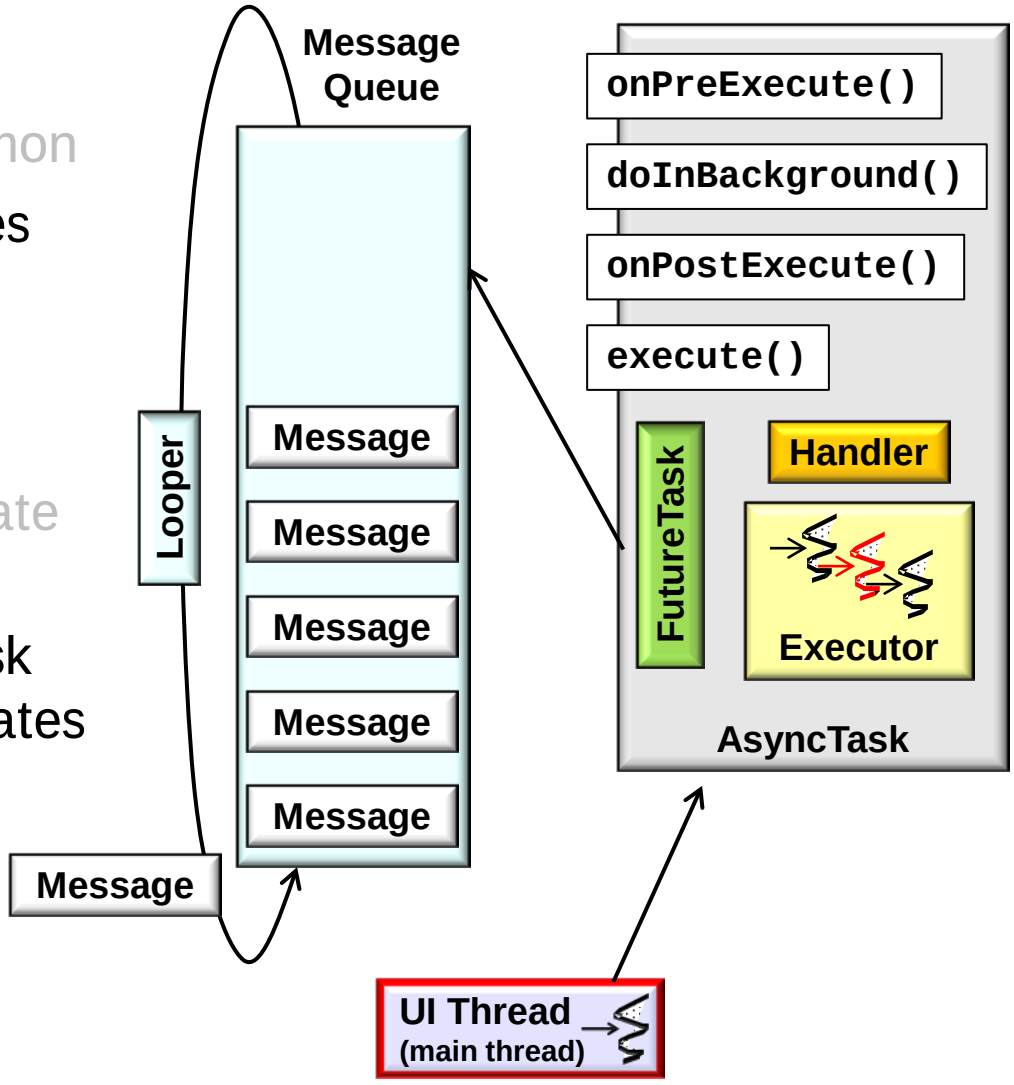
Summary

- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
  - Background Threads & UI Thread communicate in different ways
    - Background Threads in HaMeR framework explicitly communicate with UI Thread
    - Background Thread in AsyncTask framework implicitly communicates with UI Thread



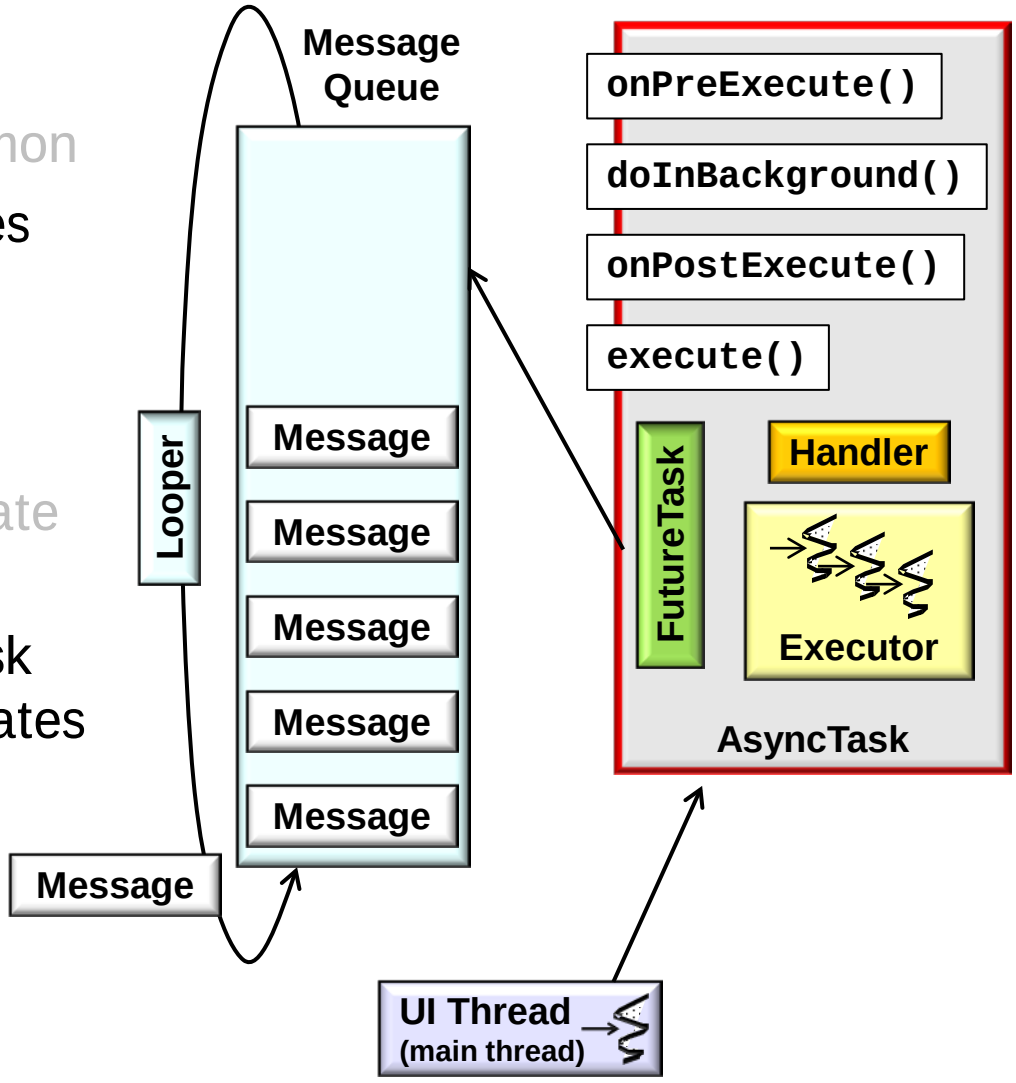
Summary

- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
  - Background Threads & UI Thread communicate in different ways
    - Background Threads in HaMeR framework explicitly communicate with UI Thread
    - Background Thread in AsyncTask framework implicitly communicates with UI Thread



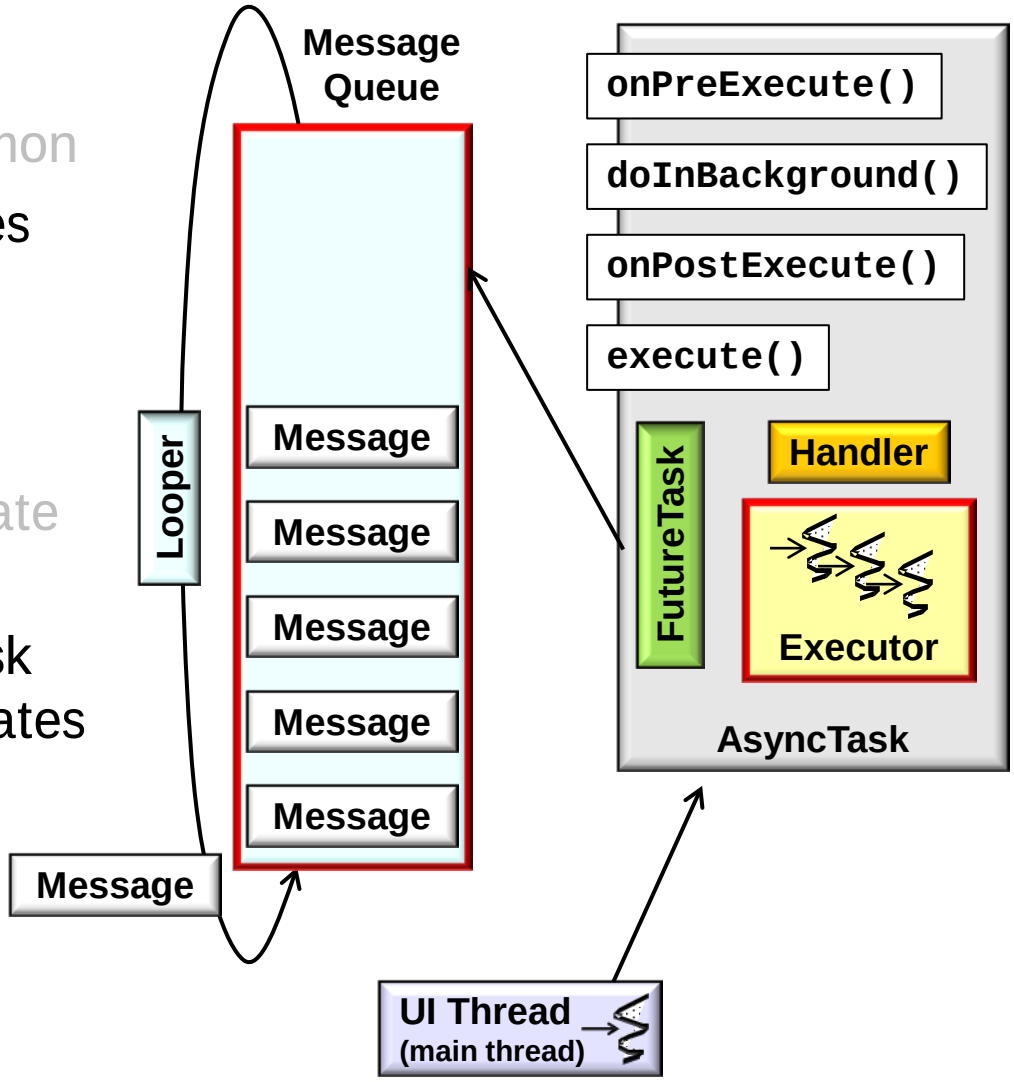
Summary

- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
  - Background Threads & UI Thread communicate in different ways
    - Background Threads in HaMeR framework explicitly communicate with UI Thread
    - Background Thread in AsyncTask framework implicitly communicates with UI Thread



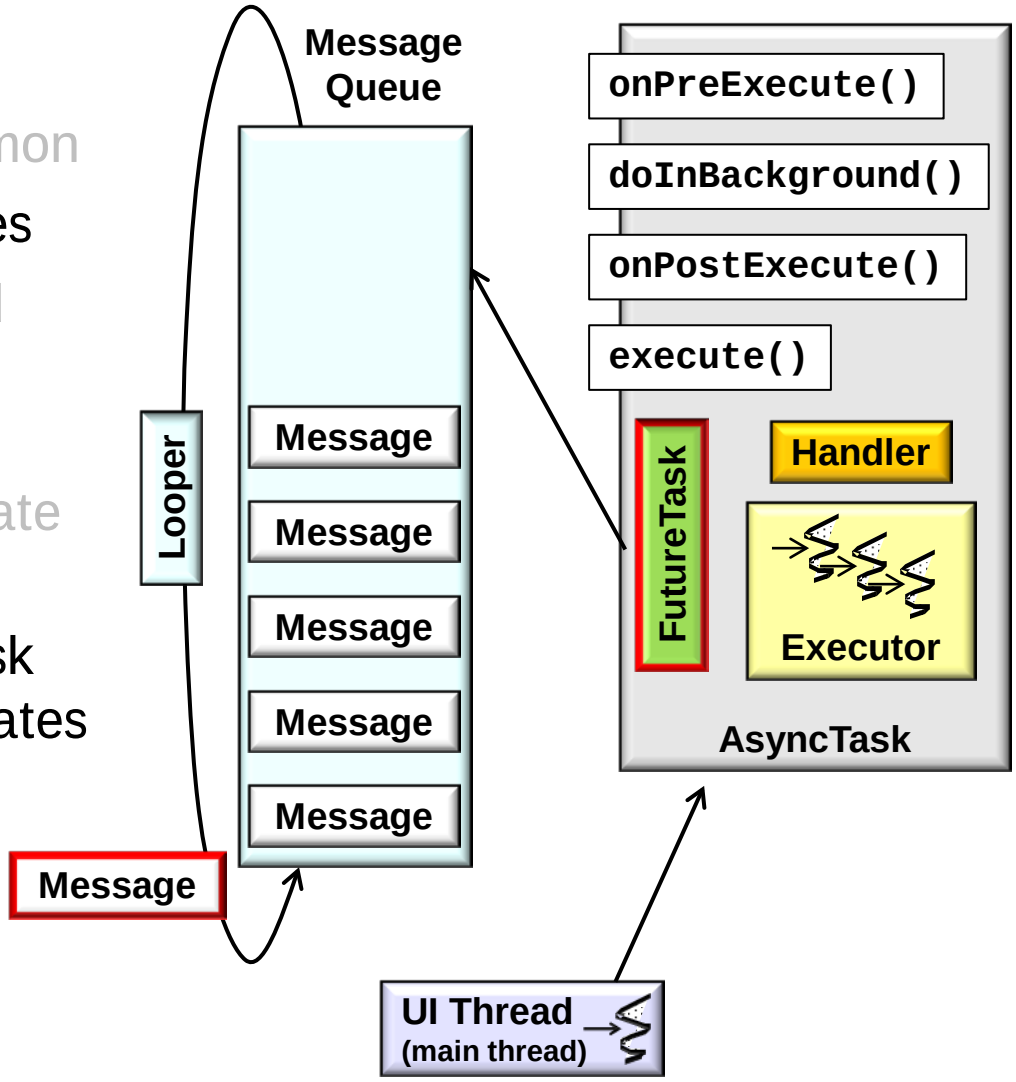
Summary

- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
  - Background Threads & UI Thread communicate in different ways
    - Background Threads in HaMeR framework explicitly communicate with UI Thread
    - Background Thread in AsyncTask framework implicitly communicates with UI Thread



Summary

- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
  - Background Threads & UI Thread communicate in different ways
    - Background Threads in HaMeR framework explicitly communicate with UI Thread
    - Background Thread in AsyncTask framework implicitly communicates with UI Thread



See earlier parts on "The AsyncTask Framework"

Summary

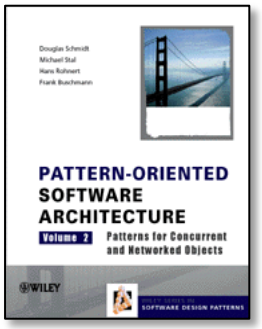
- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
- Each solution has pros & cons

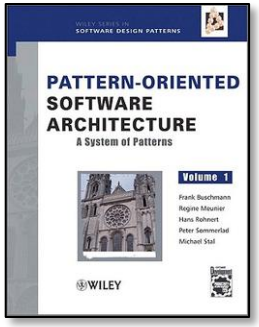
	Async Task	Posting Runnables	Sending Messages
Usability (Simple)	□□□	□□□	□□
Usability (Complex)	□□□	□	□□
Scalability	□□□	□	□
Flexibility	□□	□	□□□
Efficiency	□□	□□□	□□□

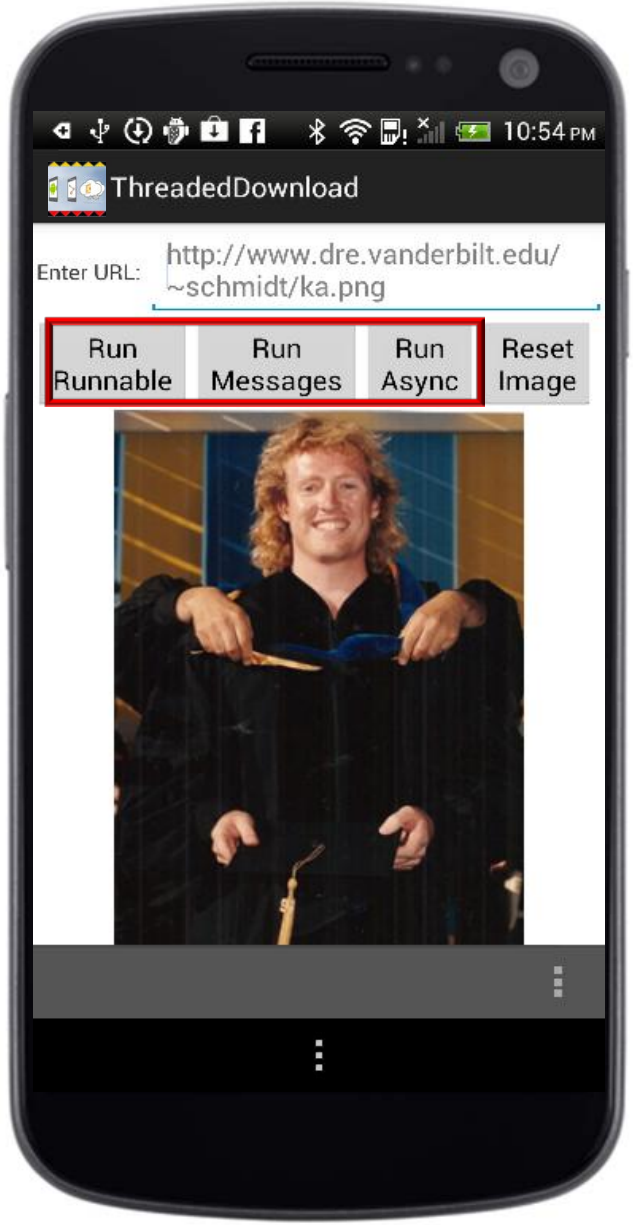
It's important to understand application requirements to make the right choice

Summary

- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
- Each solution has pros & cons
- These solutions are based on GoF & POSA patterns







Summary

- Threaded Downloads implements three different concurrency models
- Solutions share some thing in common
- Solutions also have some differences
- Each solution has pros & cons
- These solutions are based on GoF & POSA patterns