Event-Driven Asynchronous Network Programming Part 1

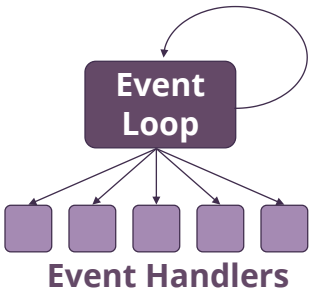
MCIT 595

Two Techniques for Handling Blocking System Calls

- **Solution I:** Multithreading
- **Solution II:** Event-Driven Asynchronous Programming

Event-Driven Programming

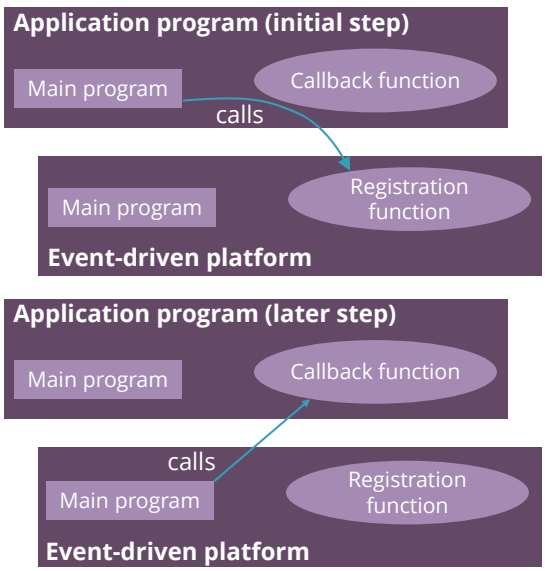
- One execution stream: no CPU concurrency
- Register interest in events (callbacks)
- Event loop waits for events, invokes handlers
- No preemption of event handlers
- Handlers generally short-lived



Callbacks and Event-loops

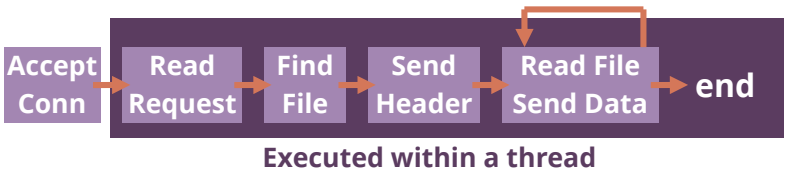
```
function read_data(fd)
  data = read_async(fd)
  if len(data) == 0
    event_polling_register(fd, read_data)
  else
    add_data_to_buffer(buffer, data)
```

- Explicit state must be saved to enable the protocol code to keep track of what to do next
- In a threaded system, the thread stack automatically stores the protocol state.

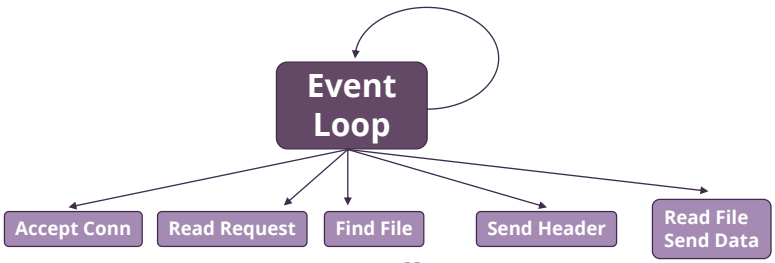


Event-Driven Web Server

- Request processing broken into separate steps
- Asynchronous read(), write(), select() for sockets



Executed within a thread



Event Handlers