Use of Tornado in KAT-7 and MeerKAT

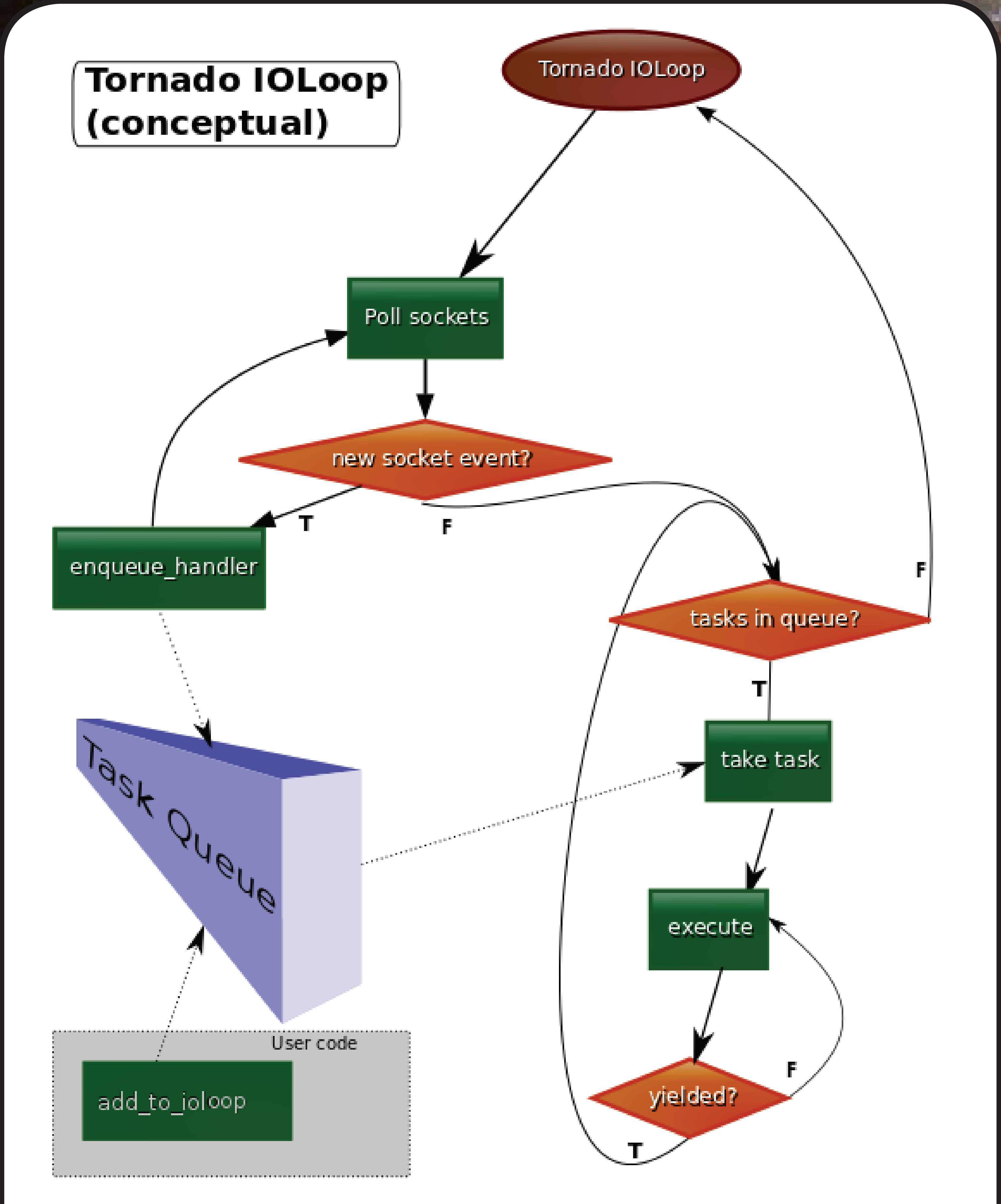
Charles de Villiers (charles@ska.ac.za),
Bulelani Xaia (bxaia@ska.ac.za)
SKA South Africa

ICALEPCS 2015
International Conference on Accelerator & Large Experimental Physics Control Systems

Background image:
NGC-101 (Pinwheel Galaxy)
Credit: Wikipedia

KAROO ARRAY TELESCOPE CONTROL PROTOCOL (KATCP)[1,2]

- Control and Monitoring (CAM) software for the Karoo Array Telescopes
- Simple textbased protocol for control and monitoring
- Used for KAT7 (prototype) now for MeerKAT
- Provides abstractions for a networked system Message, Server, Client, Sensor
- Original implementation used Python threading for concurrency



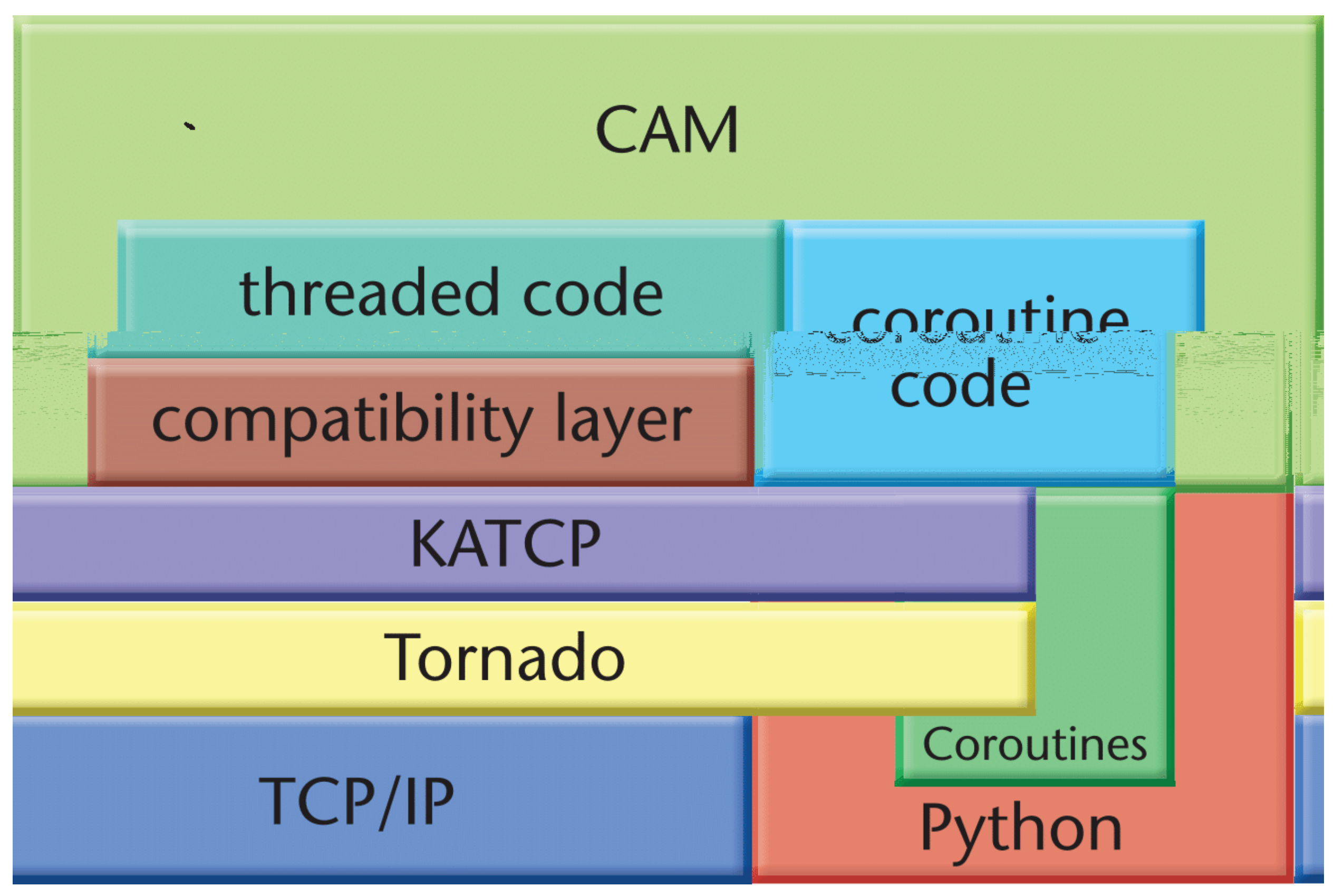
TORNADO[3]

- Concurrency framework and Web server written in Python[4]
- Highly scalable
- Supports nonblocking I/O
- Provides scheduling on top of coroutines
- Caller must 'yield' the Future if it needs the result
- Scheduler can proceed with other, nonblocking tasks

ADAPTING CAM AND KATCP TO TORNADO

- KATCP and CAM core classes have been rewritten to take advantage of Tornado coroutines
- But there is much legacy code that expects synchronous responses
- Compatibility layer (using decorators) takes care of the differences
- Clients can select a synchronous or asynchronous interface
- CAM software currently includes both types of client

CAM SOFTWARE LAYERS



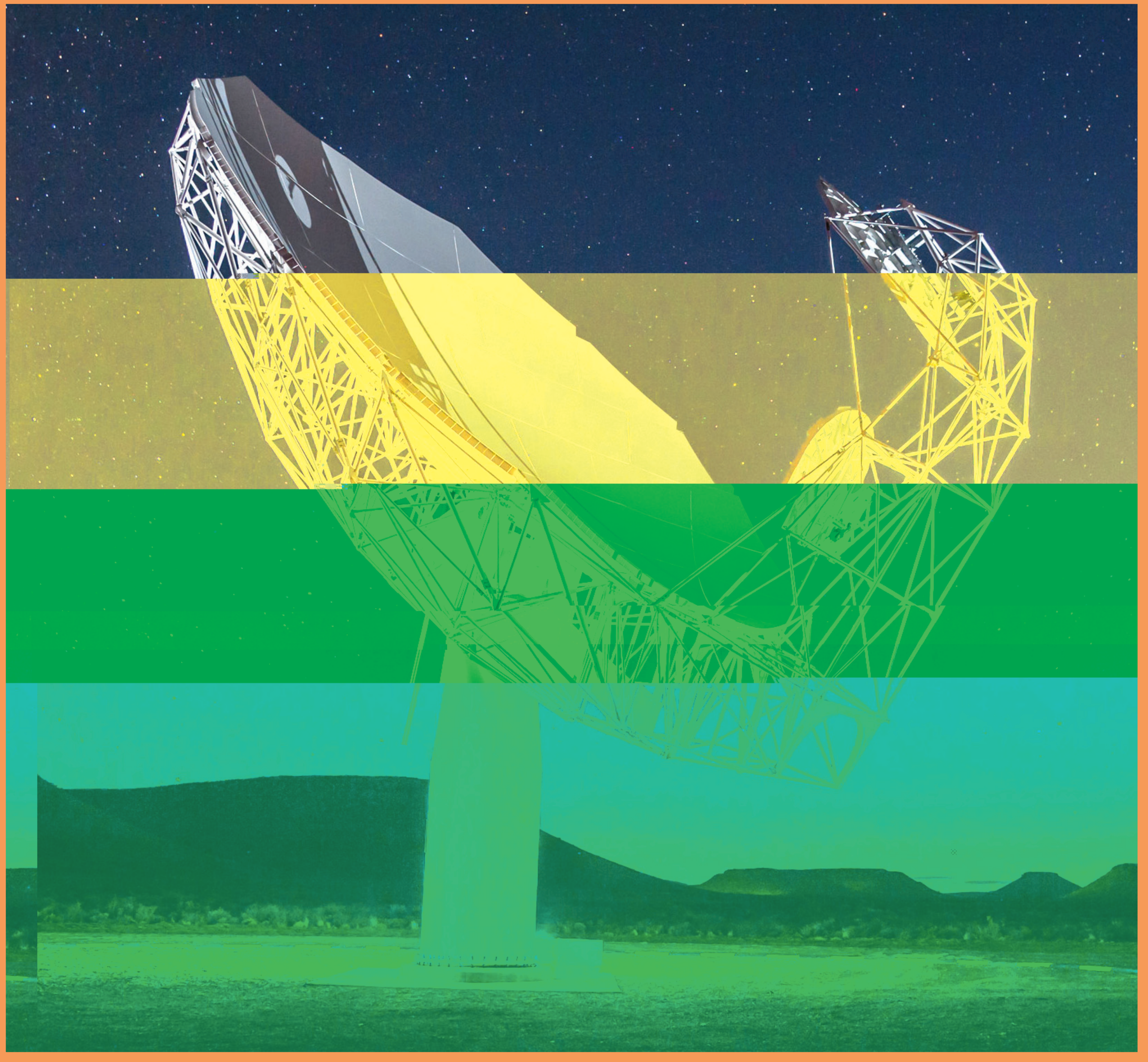
THREADS vs COROUTINES

Threads

- Directly supported by OS and Python
- Familiar to most developers
- Allow responsiveness in an I/O bound system
- Lighter than processes, but still 'heavyweight' use too many resources
- Nondeterministic behaviour depends on system scheduler
- Determinism demands complex code and careful design
- Hard to use correctly, hard to debug, hard to maintain

Coroutines

- Execute within a single thread (mostly)
- Cooperative multitasking
- Developer determines points where context may switch
- Simpler code, easier maintenance
- Support large numbers of persistent connections
- 'Lightweight' (non-OS) context switch
- Allow independent tasks to proceed without blocking
- Generally return a Future placeholder for a pending result



SUMMARY

- Tornado is starting to deliver on its promise of efficient multitasking
- The Tornado Web server and testing framework are also proving useful
- Application code simplifications are being achieved by the removal of complex locking logic
- Simpler code means better, more reliable code
- The effort of conversion has been considerable, but we believe it has been worthwhile

REFERENCES

1. KATCP documentation: https://pythonhosted.org/katcp/
2. KATCP GitHub repository: https://github.com/skasa/katcppython
3. Tornado documentation: http://tornado.readthedocs.org/en/stable/
4. Python website: https://www.python.org/