



---

# Software Development for Mobile Platforms – A Case Study

Ian Smith

[ian@abelon.com](mailto:ian@abelon.com)

# My Background



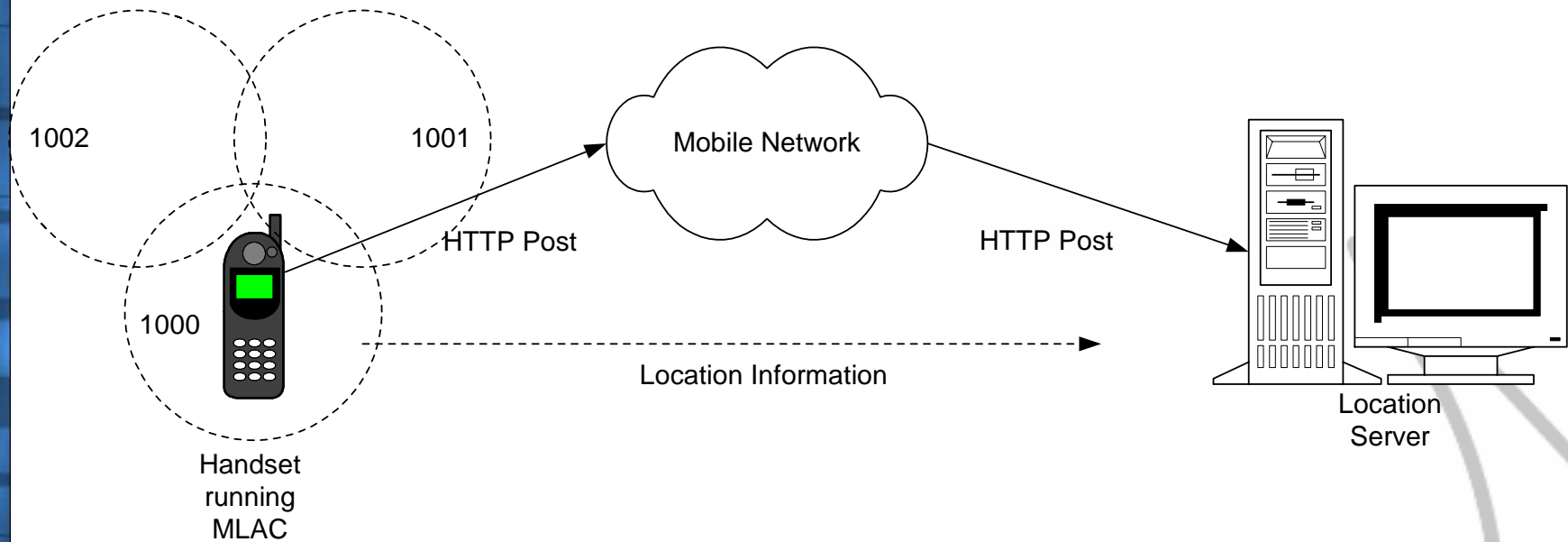
- Worked in telecomms sector since 1994
- Originally worked for Spider Systems, which then became Shiva Corporation
- Set up Edinburgh R&D office for Ascend Communications in 1998
- Technical Director for Lucent Technologies' Edinburgh R&D facility until November 2002
- Now MD of embedded systems design consultancy

# Project Description

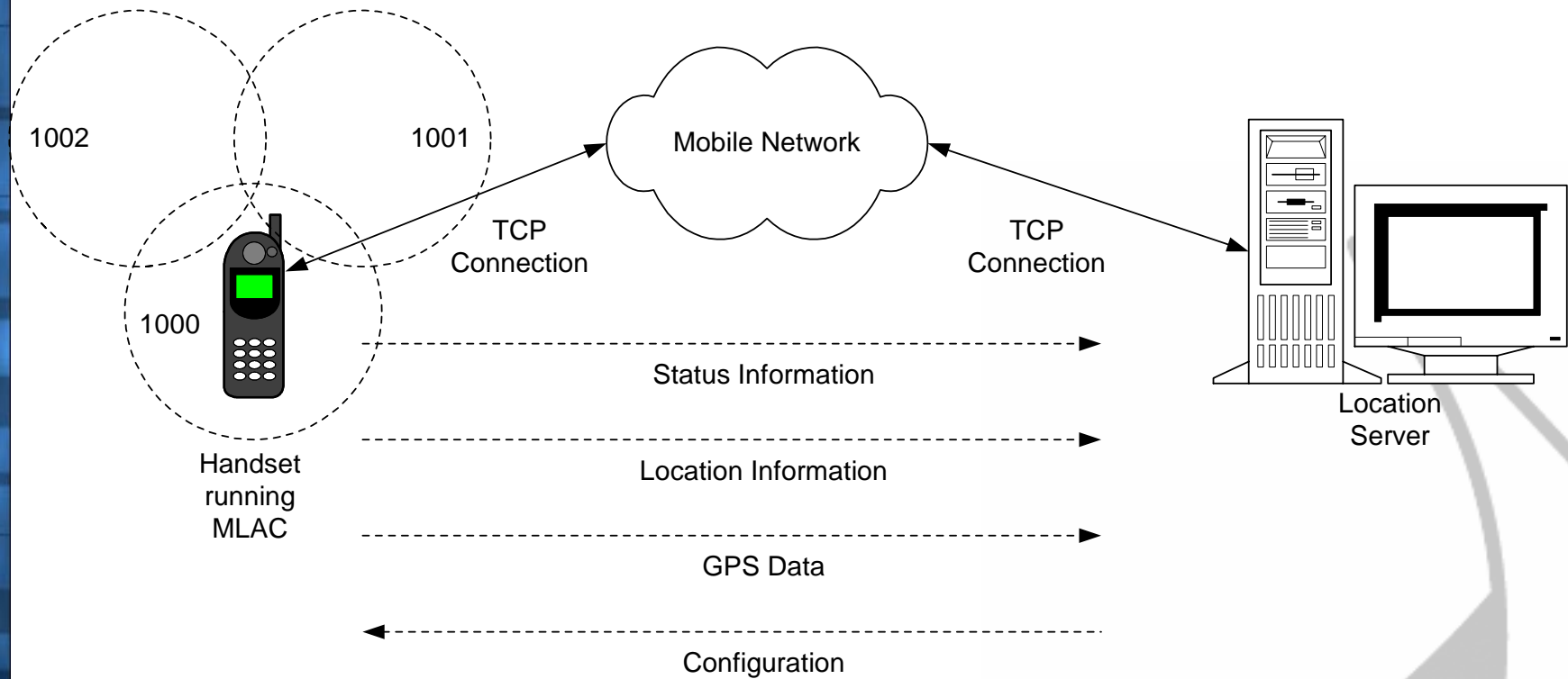


- Client had developed a prototype system – Mobile Location Aware Client (MLAC)
  - Simple system for reporting handset location to a server over GPRS
  - “GPS” functionality but using Cell handovers
  - Applications include mobile worker tracking and lone worker safety
- Our task was to “productise” this prototype
  - Improve the robustness of the communications using reliable stream-based protocol
  - Allow server-initiated commands and configuration
  - Add support for Bluetooth GPS units
  - Target platforms were Symbian S60 and J2ME

# Prototype System



# Production System



# Design Considerations



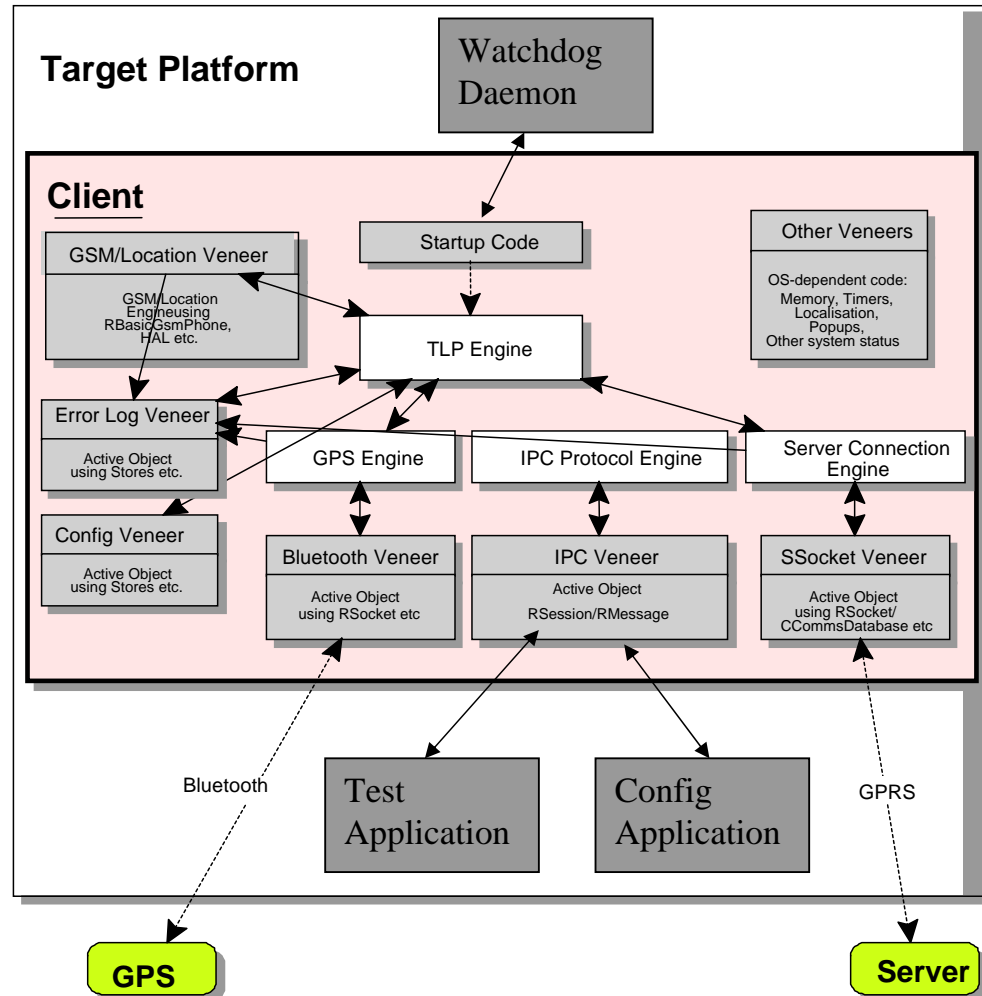
- It's a Phone!
- Code should be written in platform-independent, ISO C to assist future porting tasks.
- Platform-dependent code should be hidden behind function-based APIs (called "veneers").
- Core logic should be structured into platform-independent "engines", each having its own public 'C' function-based API
  - Code and data belonging to a given engine must only be accessed via this public API.
- All API calls shall be non-blocking.
  - All API functions must return immediately - 'blocking' is not permitted
  - A callback mechanism is used to inform requesters when processing for a long-running request has completed.
- The application protocol between Client and Server will be implemented over a reliable transport (TCP/IP).

# System Architecture



- Client has Four Core Areas of Functionality
  - TLP Engine
  - Server Connection Engine
  - IPC Protocol Engine
  - GPS Engine
- Other Functions
  - Watchdog Daemon
  - Configuration Application
  - Error Logging
  - Test Application

# Client Architecture



# Technical Considerations



- It's a Phone!
- Client runs as a background process
  - Runs as a Symbian "Server" process
  - Has no user interface as such
  - All interaction handled by helper applications
- Each Engine should have Startup and Shutdown routines
  - Memory should be allocated at start-up and freed at shutdown
  - Allocated memory should be tied to a data structure so it is freed at shutdown
    - Memory freed at other times must have its pointer set to NULL to avoid double frees
  - Shutdown routine must release all resources to avoid memory or resource leaks

# Design Architecture



- Start-up Functions
  - Legal requirements of location tracking application
  - Ensures user is aware that monitoring is taking place
  - Loads and starts Client application
- TLP Engine
  - Implements Client side of communication protocol running over TCP/IP using GPRS
  - Interacts with other engines to handle events
  - Controls sending of location information
    - After given time or after number of cell changes
  - Configured using Configuration Application

# Design Architecture



- Server Connection Engine
  - Maintains connection to central server
    - Stream based so needs start and end markers for protocol messages
  - Implements an exponential back-off algorithm if connection is not available
    - Stores error codes if connection fails or drops
  - Sends periodic keep-alives over connection
    - Most users only use consumer APN
- IPC Engine
  - Handles Inter-Process Communication (IPC)
  - Message passing interface supporting multiple connections

# Design Architecture



- **GPS Engine**
  - Only used if Bluetooth interface available and a GPS device is present
  - Periodically tries to connect to GPS unit using Bluetooth Discovery
  - Obtains and reports GPS location information using GGA fix information
  - Handles unreliable nature of Bluetooth interface and cleans up resources when connection lost or restarted

# Design Architecture



- Watchdog Daemon
  - Starts Client on handset power-on
  - Restarts Client if it crashes or problem occurs
  - Logs failure code if restart needed
- Configuration Application
  - Allows user to configure and control behaviour of Client via user interface
  - Provides a Server Connection Test and Alarm Facility
  - Reads and writes settings from configuration file
  - Written as native Series 60 Symbian application
  - Uses resource files for localisation

# Design Architecture



- Error Logging
  - Maintains log of error, warning and status messages
  - Allows Client to retrieve and upload log to server
  - Timestamps error log entries
  - Deletes older entries to limit size of log and save memory
- Test Application
  - Used to test Client
  - Can insert false Cell ID changes

# Design Architecture



- Veneers
  - Implemented using platform-dependant code
  - Hide platform specific functionality
  - Provide portable non-blocking functional APIs
  - Allow calling code to remain platform independent
  - Different versions for different platforms or OS versions e.g. Symbian 6.1 and 7.0
  - May be implemented as Symbian DLLs
    - Loaded only once to save memory
    - May not use static variables!

# Design Challenges



- Maximising the amount of portable code to ease the move to other platforms
- Differences between various handsets and OSs e.g. Symbian 6.1/7.0
- Finding S60 and J2ME handsets which would allow us access to the necessary parameters e.g. Cell ID, IMEI, MCC, MNC, LAC, RAC
- Making the client robust enough to recover when GPRS or Bluetooth connections lost
- Testing!
  - Had to wrap handset in tin foil to simulate loss of signal
  - Acceptance testing involved lots of driving around to verify reporting of Cell ID transitions

# Conclusions & Lessons Learned



- Define the requirements first and make sure you agree them with the customer
- When selecting the target hardware, check it can do everything you need before you start the project
- Design the software architecture based around the needs of your (multiple) target platforms
  - Don't write the software on a PC then try to port it to the handset
- On a mobile platform every byte counts so don't use unnecessary functionality or classes
- Test early and often
  - By the time you get to system testing it is very expensive to have to go back and re-write the code because you missed something in the design

# Thank you



- 
- Any Questions?