LWIG API Survey of implementations and considerations

Carl Williams

Consideration of the API

- Examining the implications of the constrained physical and stack environment on the API model
  - API implementation
  - API specification
  - Application developer
- API considerations document should be included as part of the Light weight Implementation Guidance suite of documentation.

Why important

- There will be API changes – both in specification and the interface between the API and the lower layers (udp, tcp, IP).
- Aiding the implementors of the API – by providing common experiences learned and recommendations of how to deal with API in Light-weight stacks.
- Understanding and Supporting the Needs of *API* Learners for these light weight stacks.
  - Don’t want to have to invent or learn a whole new way to write networking applications for these devices.

Survey the API implementation experience

- Survey seeks to collect experiences from implemention of IP stacks in constrained devices with focus on API or application impacts/considerations.
- “TinyOS” University of California Berkeley, TinyOS http://docs.tinyos.net/index.php/Main_Page
- “uIP” Adam Dunkel, Swedish Institute of Computer Science, "Adam Dunkel's uIP", http://www.sics.se/~adam/uip/index.php/Main_Page
- Others
  - Proprietary stacks with API
  - Other public domains?

API implementation

- Implementation and design of the API with respect to how applications receive, process and send packets must take into account
  - The impact on RAM usage
    - Best approaches to minimize overhead
  - The impact on throughput
    - How to minimize overhead but balance performance requirements.
  - The impact on CPU utilization
    - How to minimize tasks that require additional CPU execution time.
  - The impact on Flash
    - How to balance code size for the API (libraries, code) and applications to fit into limited Flash.
- Will the applications be well-suited to resulting API changes.

Synthesis of collection of experiences

- Here is what good, what is bad
- Benefits & consequences of varied approaches
- Scaling issues – driving toward a single recommended API
  - Scaling API from say a 8-bit micro to 32-bit micro
  - Scaling from 32K of flash to 4MB flash
  - on can be provided in the API guidance.
- Is a common API specification possible – not purpose of the initial guidance document (but possible outcome).
- API experiences that may impact applications, developers, stack writers, hardware requirements

Beginnings of Synthesis

- uIP application interface
  - event driven API model
  - Standard multi-threaded model not used
    - Consumes too much RAM and CPU processing.
- TinyOS
  - Non-blocking API
    - When application interface sends a message the routine would return immediately (before msg is sent)
    - Call-back facility notifies app when sending is done.
    - Benefit: no code runs for long periods of time; otherwise, pkt is dropped.

Next Steps

- Continue to collect implementation experiences for survey
  - Work with IPSO alliance & other implementors
  - Proprietary stacks can provide high-level guidance information on internals
- *Continue to Synthesis*
  - *Continue to update the analysis*
  - *New perspectives I have not thought about*