Lightweight TCP/IP Considerations

Lightweight IP Protocol Design BOF

David Borman – Nov 10, 2010, IETF 79, Beijing

WIND RIVER

Purpose of this presentation

- Building a light weight IP/TCP stack is about understanding the device and the applications running on it, and then optimizing the implementation to remove protocol features that are not used by the application and streamline the protocol features and supporting infrastructure that are needed.

- Strict protocol layering is a great way to design a protocol, but a terrible way to implement it.
  - paraphrased from Dave Clark

WIND RIVER

Fundamental Assumptions

- The ***protocols*** are ***not*** modified
  - Everything that is implemented must conform to the protocol specifications
- With knowledge of the application and use case, the ***implementation*** can be optimized
  - Features that are not used do not need to be implemented
    - e.g. TCP Urgent pointer
- Many parts of an implementation are not part of the protocol, e.g.:
  - Memory management
  - Connection state management
  - Timers

Operating Environment

- Hardware
  - Low power, slow CPU, small memory, low bandwidth
- Application
  - Low volume data
  - Single or small number of simultaneous connections
  - End nodes
  - Edge routers

Know the Application and Environment

- Single or multiple simultaneous TCP connections?
- Initiate or accept connections, or both?
  - Remote maintenance access
- Frequency and size of data transfer?
- Single or multi-hop network?
- End node or edge router?

Techniques - General

- Static data structures
  - Pro: pre-allocated, no need for malloc()
  - Con: hard limits
- Linked lists
  - Pro: easy to traverse
  - Con: doesn’t scale

Techniques - TCP

- Urgent pointer
  - Don’t implement it if it isn’t used, ignore on input
- MSS option
  - Should send and must process on receipt
- Receiving out of order packets
  - Can ignore them, other side will retransmit
    - Obvious performance issues
- Retransmit timer must be implemented
  - At a minimum for initial SYN or SYN/ACK

Techniques - TCP

- Single connection block
  - Pros: Only one connection to worry about
  - Cons: What do you do with the second connection request?
    - Three possibilities:
      - Abort current connection and accept new connection
      - Ignore the new connection
      - Send a RST to the new connection
    - All three have DOS issues
    - Doesn't apply if you only initiate connections
- Small, fixed number of connection blocks
  - Many of the same issues as single connection block

Other issues

- ARP
  - Treat as static once set up?
    - Upper layers can indicate when things are failing for a refresh
- IP
  - Fragmentation & reassembly
- ICMP
  - How do you handle redirects and unreachables?

- A lot of things are more easily answered if you know your applications and environment.

Summary

- Know your application and environment!
- Implementations only need to have the parts of the protocol that actually get used by the applications
- Parts that are implemented must conform to the standards
- Must be prepared to gracefully handle unimplemented features.
- A lot of the optimization work involves the software infrastructure, not the protocols themselves.