Chapter II
Cross Layer Protocol Architectures
Cross Layer Protocol Architectures

- 1 - Definition and motivation
- 2 - Architectural approaches
- 3 - Implementation approaches
- 4 - A word of caution
Definition and motivation

• Essentials of layered protocol architectures
  (Reminder)
  – Communication allowed only between adjacent layers and only via procedures calls and responses
  – Services at different layers realized by designing protocols at these layers
Definition and motivation

• Definition of cross layer design
  – Violation of the principles of layered protocol architectures
• Examples
  – Allowing communications between non adjacent layers
  – Sharing variables between layers
  – Designing protocols that span several layers
Definition and motivation

Main motivation for cross layer design

- Performance improvements, especially in wireless environments

  • An example

    - TCP sender assumes packet errors are indicators of networks congestion and slow down sending rates

      » Case of wired links: true

      » Need to slow down

      » Case of wireless links

      » Not always true

      » May be indicators of errors on physical and data link layers

      » Information from physical and data link layers to transport layer (i.e. TCP) needed to make correct decision (i.e. slow down or speed up)
Definition and motivation

Main motivation for cross layer design

– What makes wireless environments different
  • Channels vary over time and space leading to bursts of errors
    – Motion of wireless device
    – Surroundings
      » Small and large scale variations
      » Channels states can switch from good to bad within milliseconds
    » Some users may demand more channel access than others due to their location or velocity
Architectural approaches

1. Design of new interfaces
2. Merging of adjacent layers
3. Design coupling without new interfaces
4. Vertical calibration
Architectural approaches

1. Creation of new interfaces
   – Upward information flow

Ex: Explicit notifications from lower layers to TCP (e.g. explicit congestion/high error rate notification)
Architectural approaches

1. Creation of new interfaces
   – backward information flow

Ex: Applications can inform link layers about delay requirements to enable prioritization at that layer
Architectural approaches

1. Creation of new interfaces
   - Upward and backward information flow

Ex: Collaboration between physical and link layer for Collision resolution
Architectural approaches

2. Merging of adjacent layers

Rather uncommon
Architectural approaches

3. Design coupling without new interfaces

Ex: new capabilities of physical layer (e.g. possibility of receiving several packets at the same time) may trigger the redesign of a new link layer.
Architectural approaches

3. Vertical calibration

Ex: Joint tuning of parameters across the layers to achieve a specific goal
Implementation approaches

1. Direct communications between the layers
2. Shared data bases
3. New abstractions (e.g. heap)
Implementation approaches

1. Direct communications between layers
Implementation approaches

1. Direct communication between layers
   - Examples of realizations
     • Protocol headers
     • Internal packets
   - Usage / suitability
     • When few cross layers interactions are needed
Implementation approaches

2. A shared data base
Implementation approaches

2. Shared data base

- Realization
  - Quite challenging
    - Interface between layers and the data base
    - Data base structure
  
- Usage / suitability
  - Most cases, especially vertical calibration
Implementation approaches

3. New abstractions (e.g. protocol heap instead of protocol stack)
Implementation approaches

2. New abstractions
   - Realization
     • Even more challenging
       - Change the way we think about protocol implementation
   - Usage / suitability
     • A lot of potential
       - Greater flexibility
A word of caution

Benefits may not offset potential detrimental effects

– Some illustrations
  – 1. Unintended consequences
     » Tuning a parameter in layer K, to meet a specific need of layer X, may have the opposite effect on a parameter at layer B.
  – 2. “Chaos”
     » Spaghetti – like code difficult to maintain
  – Bad interactions between cross layers design
References


