Projektowanie protokołów

- Rozważmy na początek (w takiej kolejności):
  - Użycie istniejącego protokołu
  - Użycie HTTP jako platformy transportowej (?)
  - Napisanie własnej implementacji
- Protokół powinien:
  - Być jednoznaczny i bezbłędny
  - Spełniać wymagania komunikacyjne
  - Uwzględniać praktyczne warunki w jakich będzie pracował
- Czy koncepcja warstwowa może się przydać ?

Projektowanie protokołów - warstwy

- Podejście warstwowe może się przydać, jeśli chcemy zaprezentować różne poziomy abstrakcji lub definiujemy kilka rozłącznych podsystemów
- Projekt:
  - Dobrze zdefiniowane funkcje dla każdej warstwy
  - Granice warstw w punktach minimalnej interakcji
  - Minimalna liczba warstw
  - Interakcje międzywarstwowe

Projektowanie protokołów c.d.

- Rozmówca:
  - Akceptuje wszystkie prawidłowe komunikaty
  - Wykrywa każdą niepoprawną sekwencję komunikatów
  - Wydobywa się z błędów
- Realizacja:
  - Automat skończony (wejścia,wyjścia,stany)
  - Rozszerzony automat skończony (zmienne, parametry komunikatów, warunkowe zmiany stanów)

Tematy do dyskusji

- Obciążenia: procesor, łącza sieciowe itp.
- Optymalnie – czyli jak ?
- Duże rozwiązanie dla wszystkich problemów czy małe dla jednego ?
- Brak precyzyjnej specyfikacji powoduje niekompatybilne implementacje (to nie Murphy!)
- Przyszłe wersje protokołu
- Cechy protokołu: autokonfiguracja, samostabilizacja, odporność na „błędy bizantyjskie”

Radia Perlman - „Interconnections ...”

- Elastyczność, wsteczna zgodność i optymalizacje = protokół trudniejszy do implementacji = mniej popularny
- Każdy uczestnik grupy chce dodać swoją własność do protokołu
- Jeśli protokół stanie się popularny to na pewno zostanie użyty do czegoś, do czego nie był projektowany
- Protokół słabo wyspecyfikowany to „framework” (Content-type ?)

Radia Perlman - „Interconnections ...” - c.d.

•Projektuj błędy (przepełnienie buforów, przeciążenie itp.)
• Projektuj ścieżkę rozwoju dla przyszłych wersji
• Określ scenariusze dla nieznanych opcji (skip-ignore, skip-log, stop-error)
• Głównym celem jest prostota nie złożoność
• Pamiętaj o limitach i ograniczeniach sieci
• Przemyśl adresację (limitowana czy nie ?)
• DOKUMENTUJ (swój tok myślenia również ...)

Przydatne notacje

- Message Sequence Charts – diagramy podobne do diagramów sekwencji w UML
  - Ciekawy przykład narzędzia pod adresem http://www.websequencediagrams.com/

- Abstract Syntax Notation One – formalny język definicji typów