

Principles for Efficient Protocol Implementations

George Varghese

UCSD CSE

varghese@askew.wustl.edu, 314-935-4963

August 26th, 1996

2.0 Fifteen Principles for Efficient Protocol Implementation

Three categories of principles

- **Systems Principles:** 1-5 take advantage of the fact that a system is constructed from subsystems. System-wide view versus black-box.
- **Improving Efficiency while Retaining Modularity:** 5-10 improve performance while retaining modularity.
- **Speeding it Up:** 10-15 suggest techniques for speeding up a key routine considered by itself.

Amazingly, many of these principles have been used for years by Chef Charlie at his Greasy Spoon restaurant.

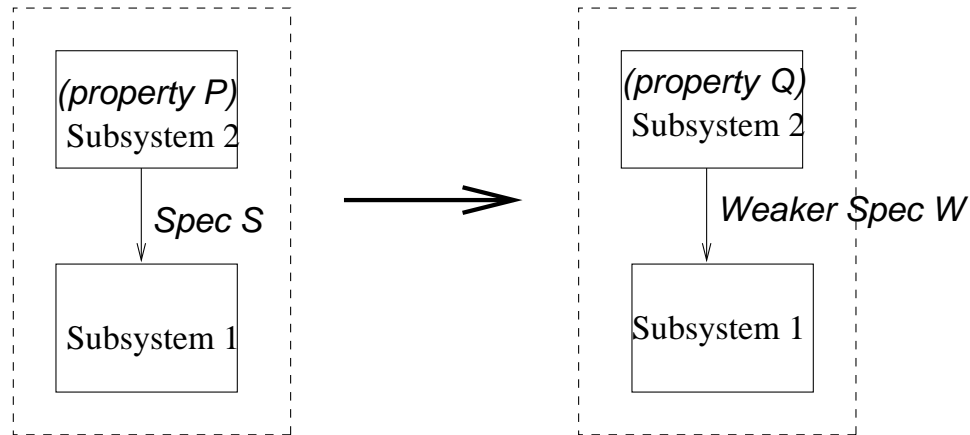
Principle 1

- **P1**, Avoid obvious waste in common sequences of operations:
- Chef Charlie: trips to the pantry to get ice-cream maker and pie dish when making pie a la mode.
- Common subexpression elimination in compilers. S1: $i = 5.1 * n + 2$ and S2: $j := (5.1 * n + 2) * 4$. Copy avoidance.

Principle 2

- **P2**, Shift computation in time:
 - *P2a*, Precompute: Prepare crushed garlic in advance, precompute TCP headers.
 - *P2b*: Evaluate Lazily: Dishwashing, copy-on-write in Mach.
 - *P2c*: Share expenses: Several pies in one oven, timing wheels

Principle 3



- **P3**, Relax specifications
- *3a, Trading Certainty for Time: Ethernet, 3b, Trading Accuracy for Time: MPEG, lossy compression, 3c, Shifting Computation in Space: IP fragmentation*

Principle 4

- P4, Leverage other system components:
- *P4a, Exploit local access costs:* Disk algorithms like B-trees.
- *P4b, Trade memory for speed:* Lookup tables, also compress to fit in cache.
- *P4c, Exploit Hardware Features:* Strength reduction in compilers.

Principle 5

- **P5**, Add hardware to improve performance:
- Microwaves for Charlie, snoopy caches for architects, content co-processors.

Principle 6

- Consider replacing unwieldy *general purpose* routines with more efficient *specialized* ones.
- Grinders for Charlie, database caching schemes.

Principle 7

- Question the need for excessive generality. If restrictions provide big gains, consider living with the restrictions.
- Exterminate Features (Thacker). RISC multiplies done on firmware. Fbufs.

Principle 8, Specification vs. Implementation

- Consider alternatives to reference implementations found in specifications as long as it has same results.
- *As an old hand, Charlie know that when a recipe asks to cut beans and then cut carrots, he can probably interchange steps without danger.*

Principle 9

- Consider passing information (that can optimize performance) between organizational layers while preserving structure.
- *Hints versus tips. Alto File System. Pointer to next file block but checked against file block number*

Principle 10

- Consider changing protocols to pass information (that can optimize performance) in protocol headers.
- *Like attention: Ms harper on letters once you know who to correspond with. Active Messages. TCP Lookups.*

Principle 11

- Optimize the expected case. Emacs buffers.
- **P11a:** Use caches.

P12, Add or Exploit State to Increase Speed

- Use of secondary indices in databases.
- Incremental Computation, strength reduction, IP checksums

P13, Optimize Degrees of Freedom

- Waiter assignments for Charlie. Ternary CAM Puzzle. Multibit tries.

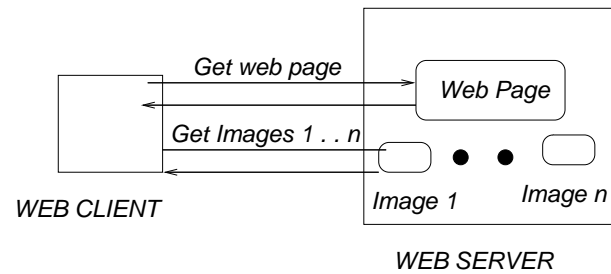
P13, Use special techniques for finite universes

- Use bucket sorting, array lookup, and bitmaps whenever possible. Page lookup and timing wheels.

Principle 15, Use algorithmic ideas

- Consider using efficient data structures that have helped protocol implementations (e.g., tries, hash tables, summary trees).
- *Charlie's recipe books have elaborate indices and crosslinks for speedy navigation.*

Caveats



Performance problems cannot be solved only through the use of Zen Meditation. (paraphrased from Jeff Mogul.

Interactions with client caching and TCP.

8 Cautionary Questions

- Q1, Is it worth improving performance?
- Q2, Is this really a bottleneck?
- Q3, What impact does the change have on the rest of the system?
- Q5, Is it worth adding custom hardware?
- Q6, Can protocol changes be avoided?
- Q7, Do prototypes confirm the initial promise?
- Q8, Will performance gains be lost if the environment changes?