Key Algorithms in a Content Delivery System

Bruce Maggs

Akamai Technologies
and
Carnegie Mellon University
Stable Marriages

- Assignment of men and women
  - Each man ranks each woman and vice versa
  - Marriage stable if no pair (m,w) unmatched where m prefers w to his "wife" and w prefers m to her "husband"

[Note: These slides are ~25 years old and if created today would likely be much less heteronormative…]
Residents-Hospitals Extension

- Residents-Hospitals
  - results + algorithm extends to case in which hospital j can accept c(j) residents
  - In use since 1951 by National Intern Matching Program
Multi-Dimensional Load

• Not a single constraining resource!
• Can be:
  - Bandwidth
  - CPU usage (e.g. key signing for https)
  - Disk usage (e.g. for cache misses, auction sites)
  - Memory (e.g. EdgeJava)
  - Threads (e.g. EdgeJava)
  - Number of licenses in Realaudio
Stable Allocations With Tree Constraints

• [G '00]:
  - resources 1,...,k
  - Supply item j has rooted tree T(j) of constraints
    • \( V(T(j)) = \{1, \ldots, k\} \)
    • Every node \( v \) of \( T \) has capacity \( c(j,v) \)
  - Demand item \( i \) has basic resource \( b(i) \) and demand \( d(i) \)
    • When \( x \) units mapped to supply \( j \), uses \( x \) units of each resource on path in \( T(j) \) from \( b(i) \) to root of \( T(j) \)
  - Stability as before
Instance of Problem

- Demand items: (groups of IPs, rule for mapping)
  \( m = \text{hundreds of thousands} \)
- Supply items: cluster of servers
  \( n = \text{thousands} \)
- (Incomplete) preference lists for demands based on performance + contract rules
- (Implicit) preference lists for supplies based on alternate choices, contract rules, …
- Tree of constraints model various resource constraints
Figure 3: An example of a resource tree with capacities shown in black. The residual capacities after assigning a video map unit with 20 units of demand requiring 20 Bps and 5 Fps is shown in blue.
Algorithm for Tree Constraints

- Demand items request unassigned demands in order of preference
- When demand i requests x units from j, repeat:
  - Find lowest (in tree) tight constraint, say node v
  - Dispose demands (up to x) of lower preference than i and using resources in subtree rooted at v