**CS262, Spring 2007 (Hellerstein/Brewer): Click**

- Number of interesting things here
  - Router via dataflow
  - Flexible and very efficient dataflow implementation
  - Some language ideas

- Routers in Dataflow
  - A bit of history.
    - Most network protocols specified in prose. Read RFC 791 some time for fun (IPv4).
    - Subsequent efforts to define crisp specification and programming languages for protocols (Estelle, Lotus, SDL) focus on Finite-State Machine descriptions
      - Computationally limited
      - Quadratic design process: messages x states.
  - One big claim of this paper is that dataflow is a nice way to program routers
    - What arguments are made? Justifications given?

- A very efficient dataflow implementation
  - elements, ports and connections
    - Even things like the scheduler are "elements", though no data ever flows through them.
  - push, pull, agnostic
    - When/why do you need push/pull? Agnostic?
      - Especially: why would you pull "deep into the plan"?
  - queues are explicit elements
    - alternatives? why is this good?
  - basics
    - single thread
    - all packet transfer/element boundary crossing is via function calls.
  - describe how control flow and dataflow are coupled
    - when does control transfer across elements?
    - when and how does data pass between elements? Where is the data?
  - scheduling
    - how/when does scheduling come into play? 3 cases!
    - discuss interrupts vs. polling
    - what is the typical unit of scheduling? can click get fancier?
    - Compare this to database iterators
  - Language ideas
    - a "wholly declarative" language
    - maybe you could prove "simple properties" of click programs?
    - method interfaces and flow-based router context
    - element granularity
      - "A Click element represents a unit of router processing".
      - small is better for flexibility
      - but when control- or dataflow doesn't match packet-flow, click needs big elements
        - the only data being passed is packets
      - some small costs in CPU (2 virtual function dispatches, can be compiled away)
CS262, Spring 2007 (Hellerstein/Brewer): Click

- ☐ where is state
  - ☐ encapsulated in elements, visible through method interfaces
- ☐ Evaluation
  - ☐ Good fit to some natural examples
    - ☐ small code
    - ☐ easy to read & modify
  - ☐ Overheads are low
    - ☐ beats handwritten, polling-based linux packet processing