Advanced Topics in Computer Systems, CS262a

Prof. Eric Brewer

Click Modular Router

I. Click Modular Router

Key ideas:

- Static graph of elements -- can verify well-formed connections
- Extenibility through interposition or redefinition -- every wire and every element is a place for possible extension
- Both push and pull processing
- Kinds of state: local, global (poor), configurations (static) and flow based
- Flexibility with only minor performance hit (mostly due to virtual functions)

Push vs. Pull:

- in/out can be push, pull or agnostic
- pull is call/return from destination, just like database query processing
- push is driven by the sender, like upcalls
- pull is good for polling and scheduling
- push is good for arrivals (and events in general)
- a queue is a push to pull converter
- Volcano's "exchange operator" is a pull to push converter, not clear if Click has an analog, although you could build one. Might be useful for multiprocessor version?

Graph properties:

- connections must match: pull to pull, push to push, or either to agnostic
- statically checked, and all agnostic ports are resolved (to push/pull) statically
- elements have configuration parameters and initilization (like "open")
- no obvious equivalent for "close", although there is an atomic update to the whole graph
- push outputs have one wire (otherwise implies duplicating output)
- pull inputs have one wire (otherwise have to decide which one to call, or merge them)
- push inputs and pull outputs can have multiple wires, first-come first served
- declarative language (for the graph) => room for optimization (see his PhD)

Scheduling:

Click Router

- single thread (!)
- only some elements need to be scheduled (tasks) -- those that push or pull independent of a caller
- most just execute when called (pushed or pulled)
- stride scheduling for tasks (why is this a good idea?)
- tasks must yield the thread by returning -- not clear how long tasks should run or how it affects future scheduling decisions
- deadlock? likely if any element can block, since there is no pre-emption and no other threads. exception: block for arrival might work (since that is a form of concurrency)
- Polling vs. Interrupts: polling is much faster (8x?); processor stays in the sweet spot (no pipeline flushes), better caching, and easy to batch arrivals for lower overhead (and even more locality)
- "receiver livelock" -- problem in which interrupts are arriving faster than the handling rate, which prevents any real work from getting done. Polling doesn't have this problem, but you must drop packets at poll time to keep it in the operating range. Note how flat the throughput curves are when the reach saturation -- excellent "graceful degradation", much like SEDA
- Note the use of shared memory to get packets, much like U-Net.

Flow Context:

- Queried at initialization time to figure out properties about your context (neighbors)
- Example properties: nearest queue, how many queues downstream or upstream, what interface is this from
- Enables elements to behave differently based on context
- Best example is looking at the total downstream queue length for congestion estimation
- Side note: RED is "random early detection" (Floyd et al. 93). Key idea is to drop packets with (linearly) higher probability as the queue length gets longer, starting at some min. Attacks congestion proactively and earlier, hoping to keep the delays low.
- In a DBMS, these kinds of things are done by the query optimizer, which typically has lots of specialized operators (and parameters) to use in different contexts

Initialization:

- broken into stages to avoid cycle dependencies
- not clear who decides what goes in each stage; presumably the programmer has to participate

Annotations:

- Idea: add metadata to packet to carry information downstream
- Statically defined fields

- In theory every producer of an annotation should have exactly one consumer on all possible paths (where discard counts as a consumer). This does not appear to be enforced (or enforcable), since the annotations are not part of the language.
- This is a bit of a hack: it is its own system that can't be type checked or verified, but it still required for correct operation.
- Also no support for global information. Not clear why this was avoided. (In general, larger scopes are bad, but if you need one, it is better to create one than to abuse other mechanisms like configurations or annotations.)

Peformance:

- overall excellent
- polling really helps
- flexibility costs about 1us per packet -- mostly due to virtual functions (in C++). This is because every element is a subclass with lots of virtual functions, so every push/pull call is a virtual function call with a dynamic level of indirection. However, since the graph is really static, you can flatten the virtual functions either statically or (in theory) at initialization time.
- Supports its claims of extensibility and flexibility quite well
- Supports its claim of low overhead for this flexibility