**Turbo**

* Need ultra-low latency remote procedure calls.
  + 5-10 us in short term; 1-2 us in long term (RAMCloud)
  + 2us x 10Gbps = 2.5KB = Bw-delay product
  + RDMA
* **Initial Approach**: We started by taking trying to remove buffering from the network.
  + Advantages:
    - Predictable delays per packet
    - Finer grained retransmissions (reduces latency if loss is kept low (?))
    - Cost
    - Simplicity of design – Priority scheduling and dropping are O(1).
* Question: What network protocol to use?
* Looked at how existing approaches perform
  + TCP:
    - Works quite well for some flows
    - But really poorly for others
    - Very low utilization – Could only sustain load-factor of 0.4
    - Tried modifications to TCP with packet jittering with similar results

Macintosh HD:Users:gautamk:Dropbox:Work:Hawthorn:TCPNewReno:Results:10-29-2013:PercentageIncreaseInFct.pdf

* + State of the art: pFabric:
    - Analyzed its performance when buffers are very small
    - It performs poorly (graph to be redrawn)
* Looking from a different perspective:
  + pFabric claims near-optimal performance (though with non-trivial buffering, 10x the bw delay product we aspire to achieve).
  + Issues
    - Can cause starvation – a plot of normalized CDF of pFabric shows that flows can be 100x from the ideal and many of these are long flows but a better experiment to be done; Kaifei helping with this)

Macintosh HD:Users:gautamk:Dropbox:Work:Turbo:Results:Overhead:NormalizedFctCdf.pdf

* + - Dead packets can cause suboptimal utilization
    - Quite far from optimal using a sane metric – comparing averages of two systems
    - (Secondary) Complicated design state for every flow (to select the earliest packet from the highest priority flow)
  + Overhead that pFabric induces
    - pFabric adds around 18% byte overhead in terms of the packets that get delivered to the NIC
    - But many of these packets get dropped at the NIC and the network overhead is 5%.
* **Current approach**
  + Formalize the global optimization problem (High Priority)
  + But we want a distributed solution
  + Break up the global optimization problem into locally executable chunks
    - E.g., utilization maximization by making sure that a packet that has already traversed (n-1) hops also gets scheduled on the last hop to prevent bandwidth wastage.
  + Compose the local (or separate) chunks to approximate the global solution
  + Our current mindset is to heavily leverage priorities – using ideas from OS techniques to assign scheduling priorities based on a number of ideas (e.g., Lottery scheduling idea implemented by making retransmissions at a higher priority). [More thought required]
  + This gives us a k-tuple of priorities; the theoretical question would be how to compose these priorities to get a single number to get close the global optimal.