Queueing in switched networks

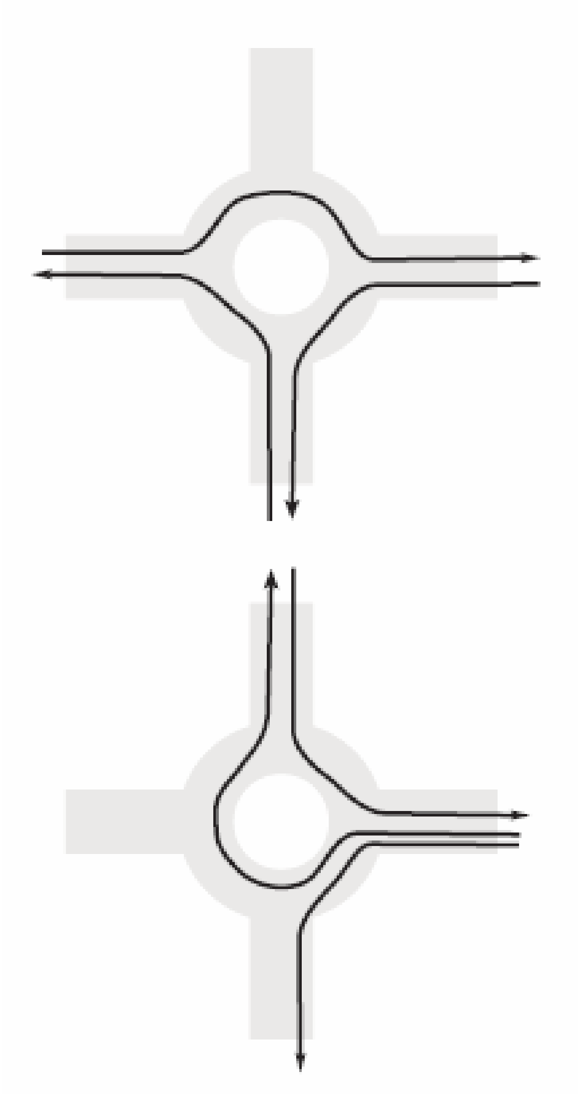
Damon Wischik, UCL

thanks to Devavrat Shah, MIT

Outline

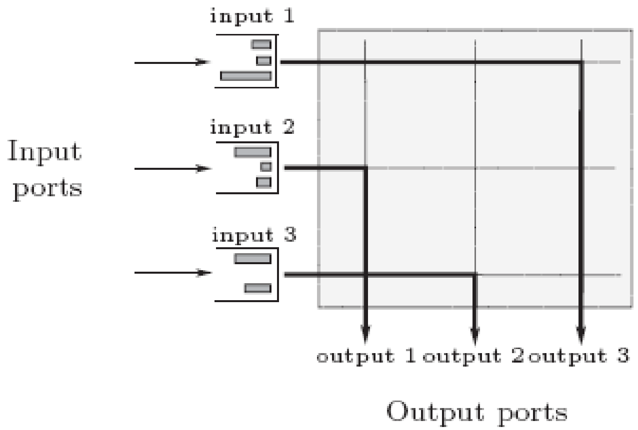
- Applications and illustrations
- The static planning problem and its dual
- What we want to answer
- Examples of scheduling algorithms
- Analysis
- Designing practical algorithms

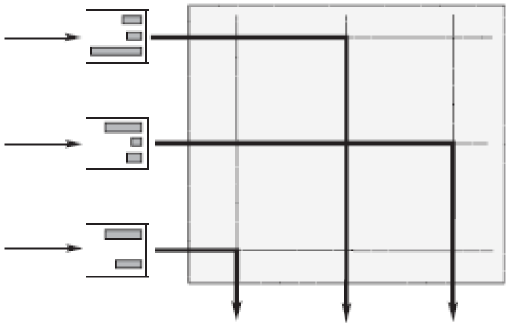
Roundabouts and crossroads



- There are twelve possible flows of traffic
- The road layout places constraints on which flows can use the roundabout simultaneously
- Traffic regulations, and maybe traffic lights, determine which flows get to move

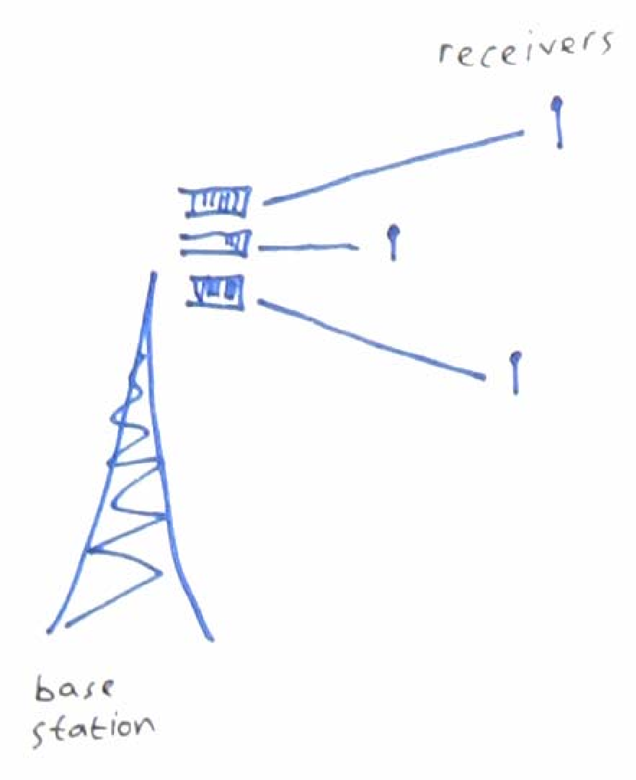
Input-queued switches





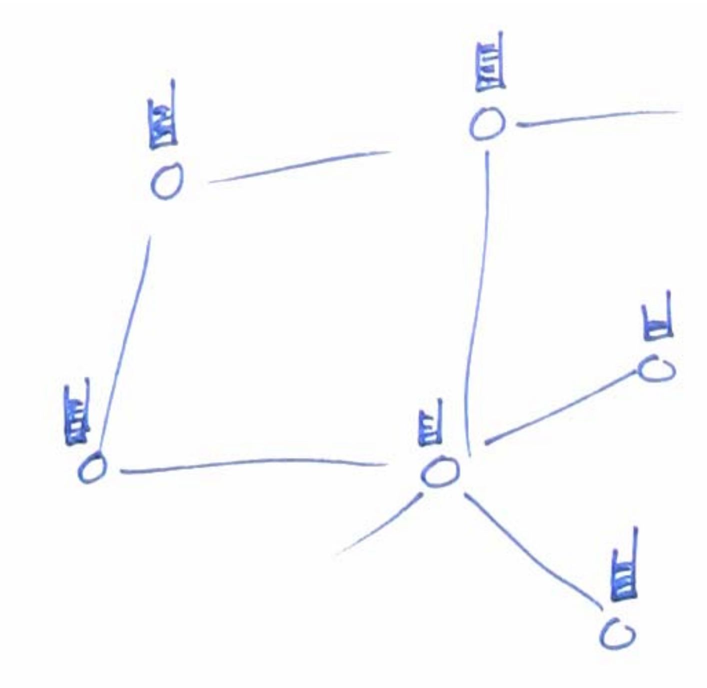
- This is a model for the silicon fabric at the core of a high-speed Internet router
- Slotted time, fixed-size packets
- At each timeslot, the switch chooses a matching of inputs to outputs, and serves the corresponding queues

Wireless base-station



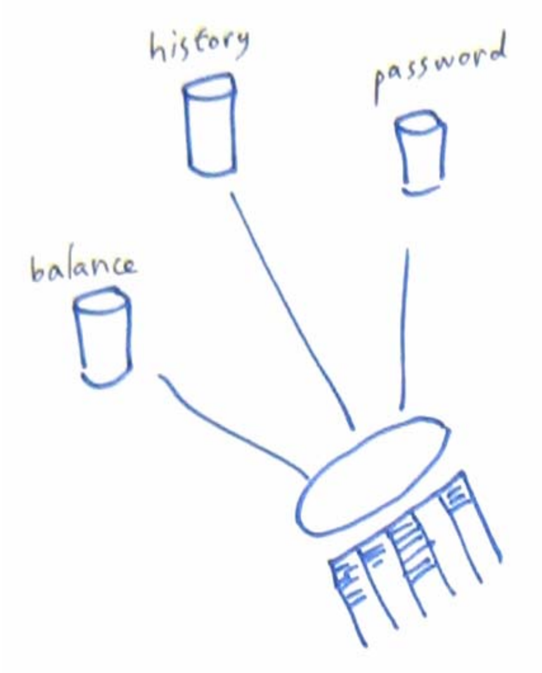
- A wireless base station, transmitting data to several users
- At each timeslot, the base station chooses what power to use to transmit to each of the users
- The resulting transmission rates depend on interference, distance, and channel state

Wireless ad-hoc network



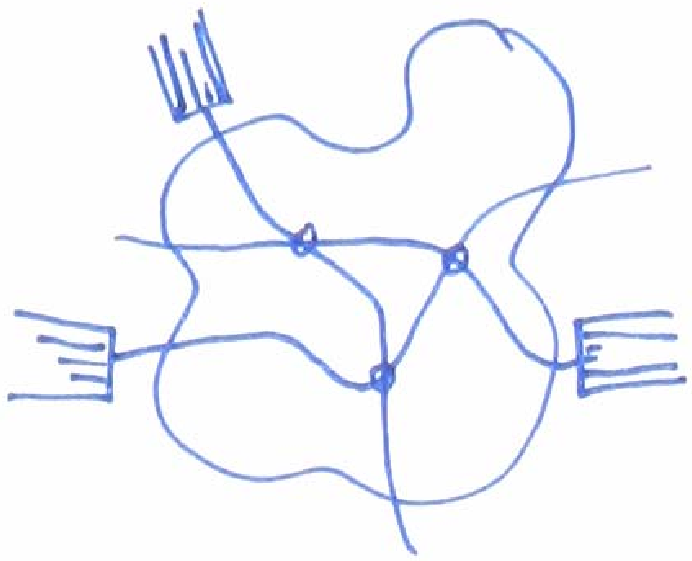
- Each node has a stream of data to send to its neighbours
- Each node can broadcast to its neighbours; if a node receives more than one broadcast, both are lost
- Each node can choose a broadcast probability, and every timeslot it broadcasts with this probability
- These choices determine the throughput

Database concurrency control



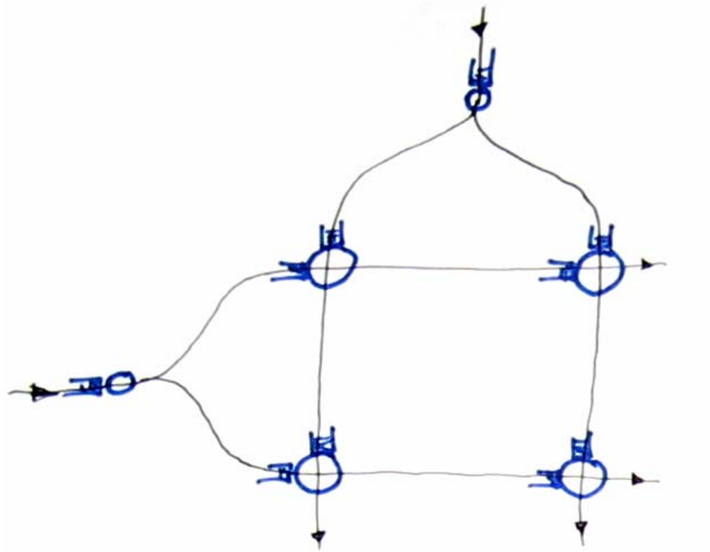
- A system runs several databases, and it receives a stream of jobs
  - Each job may require *write access* to some of the databases, and *read access* to others
  - If a job is writing to one of the databases, no other jobs can read from that database at the same time
- The system chooses which jobs to run concurrently

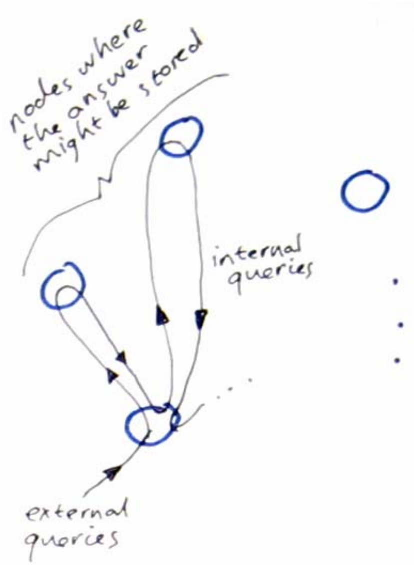
Flow-level model of TCP



- Consider a set of routes through the Internet
- There may be a number of simultaneous TCP flows on each route
- TCP determines the transmission rate that each flow receives, given the numbers of jobs and the routes
- This has the effect of draining the queues of jobs

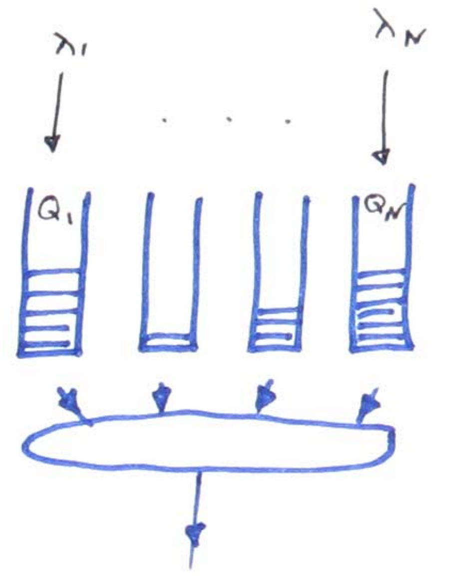
Routing and sequencing in a multihop network





- An abstract queueing network with routing and sequencing choice
- A DHT. A query arrives at a node, and is forwarded to a succession of other nodes until it finds an answer
  - nodes may choose to drop a request if they are overloaded
  - they also choose which queue to serve

Abstract model



- A single-hop network (packets leave once they are served)
- Slotted time, equal-sized packets
- N queues, with the vector of queue sizes $\boldsymbol{Q}(t) = \big(Q_1(t), \ldots, Q_N(t)\big)$
- An exogeneous arrival process of rate $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_N)$ to each queue
- Each timeslot, an action $\boldsymbol{\pi}(t)$ is chosen from a finite set $\mathcal{S} \subset \{0, 1\}^N$

The static planning problem

PRIMAL($\boldsymbol{\lambda}$)

minimize $\sum_{\boldsymbol{\pi} \in \mathcal{S}} \alpha_{\boldsymbol{\pi}}$ such that $\boldsymbol{\lambda} \leq \sum_{\boldsymbol{\pi} \in \mathcal{S}} \alpha_{\boldsymbol{\pi}} \boldsymbol{\pi},$ over $\alpha_{\boldsymbol{\pi}} \geq 0$ for all $\boldsymbol{\pi} \in \mathcal{S}$

DUAL($\boldsymbol{\lambda}$)

maximize $\boldsymbol{\xi}^\top \boldsymbol{\lambda}$ such that $\boldsymbol{\xi}^\top \boldsymbol{\pi} \leq 1$ for all $\boldsymbol{\pi} \in \mathcal{S},$ over $\boldsymbol{\xi} \geq 0$

STABILITY REGION

$$\Lambda = \left\{ \boldsymbol{\lambda} \in \mathbb{R}_+^N \ : \text{solution to } \text{PRIMAL}(\boldsymbol{\lambda}) \leq 1 \right\}$$

The static planning problem

PRIMAL($\boldsymbol{\lambda}$)

$$\text{minimize} \sum_{\boldsymbol{\pi}\in\mathcal{S}} \alpha_{\boldsymbol{\pi}} \text{ such that } \boldsymbol{\lambda} \leq \sum_{\boldsymbol{\pi}\in\mathcal{S}} \alpha_{\boldsymbol{\pi}}\boldsymbol{\pi}, \text{ over } \alpha_{\boldsymbol{\pi}} \geq 0 \text{ for all } \boldsymbol{\pi} \in \mathcal{S}$$

DUAL($\boldsymbol{\lambda}$)

maximize $\boldsymbol{\xi}^\top \boldsymbol{\lambda}$ such t

STA

$\Lambda =$

Interpret α_π as the fraction of ti
which we should run schedule

If the solution to this problem

Imagine that each packet in queue *n* is worth ar
money ξ_n. Then $\xi^\top\lambda$ is the total rate at which m
$\xi^\top\pi$ is the most money that action π can tak

If the solution to this problem is >1, then the am
the system will build up, i.e. the queue lengths w

For $\lambda \in \Lambda$, it is possible to schedule all incoming work and keep the system stable. Otherwise the system is unstable.

Interesting questions

- Given a scheduling algorithm, is it stable for all $\lambda \in \Lambda$? (described as *having 100% throughput*)
- Is there an online algorithm, i.e. one whose action in timeslot *t* depends only on queue sizes at timeslot *t*, which has 100% throughput?
  - such an algorithm ought to be more responsive to transient conditions
- Even if an algorithm has 100% throughput, it may have terrible performance. What are the properties of an algorithm which lead to low average delay?
- If the network is overloaded, i.e. PRIMAL(λ)>1, does the algorithm work OK?
  - e.g. maximize net departure rate

Some scheduling algorithms

- BIGSTEP
  - Count arrivals over T slots. Compute a sequence of schedule which would serve them. Use these schedules for the next T slots.
- Greedy
  - Serve the biggest queues you can
- MaxSize
  - Pick any schedule which maximizes the number of departures
- MaxWeight
  - pick any schedule π which maximizes $\pi^\top Q$
  - or, pick any schedule π which maximizes $\pi^\top(Q^\alpha)$, for some prespecified $\alpha>0$, where the exponent is componentwise

Analysis I. Foster-Lyapunov criteria

Let X_n, $n \in \mathbb{N}$, be an irreducible aperiodic Markov chain which takes values in a countable state space $\mathcal{X}$. Suppose

(i) $|X_{n+1} - X_n| \le f(X_n)$ almost surely, for some finite-valued function $f(\cdot)$

(ii) $H : \mathcal{X} \to \mathbb{R}_+$ is some function with finite level sets, i.e. $\{x : H(x) \le \theta\}$ is finite for all levels $\theta \in \mathbb{R}_+$

(iii) $L : \mathcal{X} \to \mathbb{R}_+$ is another function with finite level sets

Theorem. If there exist constants $\varepsilon > 0$ and $B \ge 0$ such that

$$\mathbb{E}\big[L(X_{n+1}) - L(X_n) \;\big|\; X_n\big] \le B - \varepsilon H(X_n)$$

then X_n has a unique invariate distribution, and

$$\limsup_{n\to\infty} \mathbb{E}[H(X_n)] \le B/\varepsilon.$$

Applying the Foster-Lyapunov drift criterion

Let $L(\boldsymbol{Q}) = \sum_n Q_n^2 = \boldsymbol{Q}^\top \boldsymbol{Q}$.

$$
\begin{aligned}
L\Big((\boldsymbol{Q}(t+1)) - L(\boldsymbol{Q}(t)\Big) &= \Big(\boldsymbol{Q}(t+1) - \boldsymbol{Q}(t)\Big)^\top \Big(\boldsymbol{Q}(t+1) - \boldsymbol{Q}(t)\Big) \\
&= \boldsymbol{\Delta}(t)^\top \Big(2\boldsymbol{Q}(t) + \boldsymbol{\Delta}(t)\Big) \quad \text{where } \boldsymbol{\Delta}(t) = \boldsymbol{Q}(t+1) - \boldsymbol{Q}(t) \\
&= \sum_n \Delta_n(t)^2 + 2\sum_n \Delta_n(t) Q_n(t) \\
&= \sum_n \Delta_n(t)^2 + 2\sum_n \Big(A_n(t) - D_n(t)\Big) Q_n(t) \quad \text{where } A_n(t) \text{ is arrivals and } D_n(t) \text{ is departures at } t \\
&= \sum_n \Delta_n(t)^2 + 2\sum_n \Big(A_n(t) - \Pi_n(t)\Big) Q_n(t) \quad \text{where } \Pi_n(t) \text{ is service, since we only fire a blank if } Q_n(t) = 0 \\
&\leq N + 2\sum_n \Big(A_n(t) - \Pi_n(t)\Big) Q_n(t) \quad \text{since } \Delta_n(t) \in -1, 0, 1 \text{ assuming Bernoulli arrivals} \\
&= N + 2\Big(\boldsymbol{A}(t)^\top \boldsymbol{Q} - \max_{\boldsymbol{\rho}\in\mathcal{S}} \boldsymbol{\rho}^\top \boldsymbol{Q}\Big) \quad \text{since matching is chosen to have max weight}
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}\Big[L(\boldsymbol{Q}(t+1)) - L(\boldsymbol{Q}(t)) \,\Big|\, \boldsymbol{Q}(t)\Big] &\leq N + 2\Big(\boldsymbol{\lambda}^\top \boldsymbol{Q} - \max_{\boldsymbol{\rho}\in\mathcal{S}} \boldsymbol{\rho}^\top \boldsymbol{Q}\Big) \\
&\leq N + 2\Big(\sum_{\boldsymbol{\pi}\in\mathcal{S}} \alpha_{\boldsymbol{\pi}} \boldsymbol{\pi}^\top \boldsymbol{Q} - \max_{\rho\in\mathcal{S}} \boldsymbol{\rho}^\top \boldsymbol{Q}\Big) \quad \text{with } \alpha_{\boldsymbol{\pi}} \text{ as in the primal problem} \\
&= N + 2(\sum \alpha_{\boldsymbol{\pi}} - 1) \max_{\rho\in\mathcal{S}} \boldsymbol{\rho}^\top \boldsymbol{Q} \\
&\leq N - \varepsilon \max_{\boldsymbol{\rho}\in\mathcal{S}} \boldsymbol{\rho}^\top \boldsymbol{Q} \quad \text{where } \varepsilon = 2(1 - \sum \alpha_{\boldsymbol{\pi}}) > 0, \text{ assuming } \lambda \in \Lambda^\circ
\end{aligned}
$$

Analysis II. Fluid stability

Consider a sequence of switched queueing networks indexed by $r \in \mathbb{N}$. Let

$\boldsymbol{Q}^r(t)$ = queue size vector at time $t, t \in \mathbb{N}$

$\boldsymbol{A}^r(t)$ = total arrivals up to time $t, t \in \mathbb{N}$

$S_{\boldsymbol{\pi}}(t)$ = number of timeslots in which $\boldsymbol{\pi}$ has been done, up to $t, t \in \mathbb{N}$

$x^r(t) = \big(\boldsymbol{Q}^r(rt)/r, \boldsymbol{A}^r(rt)/r, [S^r_{\boldsymbol{\pi}}(rt)/r]_{\boldsymbol{\pi}\in\mathcal{S}}\big), t \in \mathbb{R}$

Definition. Let FLP $= \{x$: there is a subsequence of points in the sample space, satisfying SLLN, with $x^{r_k} \to x$ in an appropriate sense $\}$

Definition. Say the fluid system is stable if there is some $T > 0$ such that $x(t) = 0$ for all $t > T$, for all $x \in$ FLP with $|x(0)| \leq 1$.

Theorem. Consider a Markov chain describing the switched queueing network, assuming IID Bernoulli arrivals. If the fluid system is stable, then the Markov chain has a unique invariant distribution.

Applying the fluid method

The typical use of the fluid method is like this. Following Dai+Prabhakar, we first prove

Lemma. If $x \in$ FLP then x satisfies all the following *fluid model equations*.

i. x is absolutely continuous, hence differentiable almost everywhere
ii. $\boldsymbol{a}(t) = \boldsymbol{\lambda} t$
iii. $\sum_{\boldsymbol{\pi}} s_{\boldsymbol{\pi}}(t) = t$
iv. $\dot{q}_n(t) = \lambda_n - \sum_{\boldsymbol{\pi}} \dot{s}_{\boldsymbol{\pi}}(t)\pi_n$, or the positive part of this expression if $q_n(t) = 0$
v. $\dot{s}_{\boldsymbol{\pi}}(t) = 0$ if $\boldsymbol{\pi}^\top \boldsymbol{q}(t) < \max_{\boldsymbol{\rho}} \boldsymbol{\rho}^\top \boldsymbol{q}(t)$

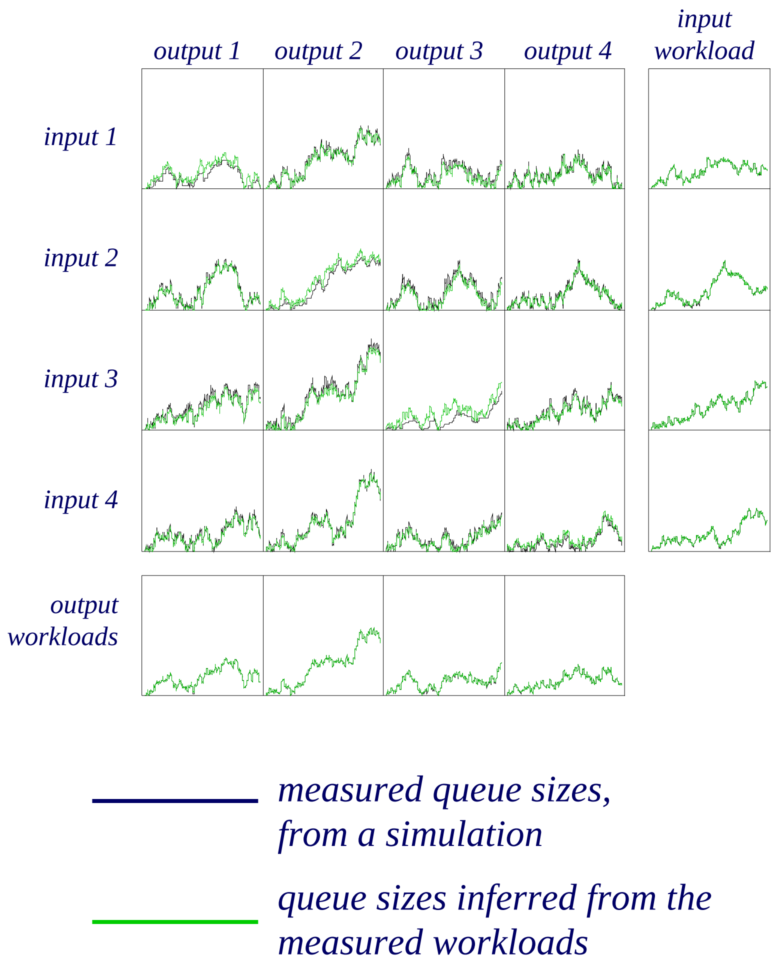
Then define FMS $= \{x : x$ satisfies all these equations $\}$. Then, using much the same reasoning as for the Foster-Lyapunov drift condition, prove

Lemma. If $x \in$ FMS and $\boldsymbol{\lambda} \in \Lambda^\circ$, then

$$\dot{L}(\boldsymbol{q}(t)) \leq -\varepsilon \max_{\boldsymbol{\rho} \in \mathcal{S}} \boldsymbol{\rho}^\top \boldsymbol{q}(t).$$

Finally, using a result of Stolyar, we obtain stability.

Analysis III. Heavy traffic



- For an input-queued switch running MaxWeight, simulations suggest that

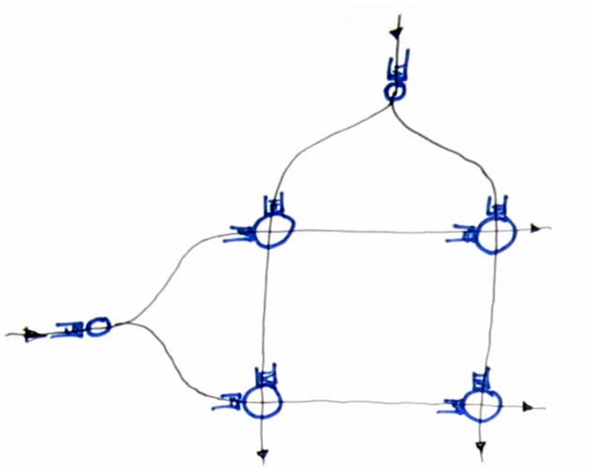
$$Q(t) \approx \Delta(w(Q(t)))$$

for suitably-chosen functions Δ and w

- This is called state space collapse, and is a general feature of heavily-loaded systems
  - that is, systems where the solution to PRIMAL(λ) is $\approx$1

- It may help us understand queueing delay for scheduling algorithms

Practical algorithms: backpressure



- MaxWeight rule says

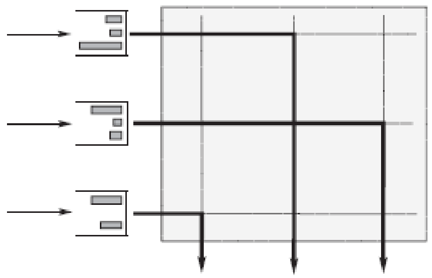
Each action $\boldsymbol{\pi}$ serves a collection of packets $p \in P(\boldsymbol{\pi})$. Each of these packets will be removed from a queue $\mathrm{src}(p)$ and sent to another queue $\mathrm{dest}(p)$, or it will leave the network.

Choose the schedule $\boldsymbol{\pi}$ which maximizes the weight

$$\sum_{p \in P(\boldsymbol{\pi})} \left[Q_{\mathrm{src}(p)} - Q_{\mathrm{dest}(p)} 1_{p \text{ doesn't leave}} \right]$$

- To compute weights, each node needs to know its queue sizes and downstream queue sizes — nothing more
- Assume that you always have the option of idling. Then MaxWeight has 100% throughput.

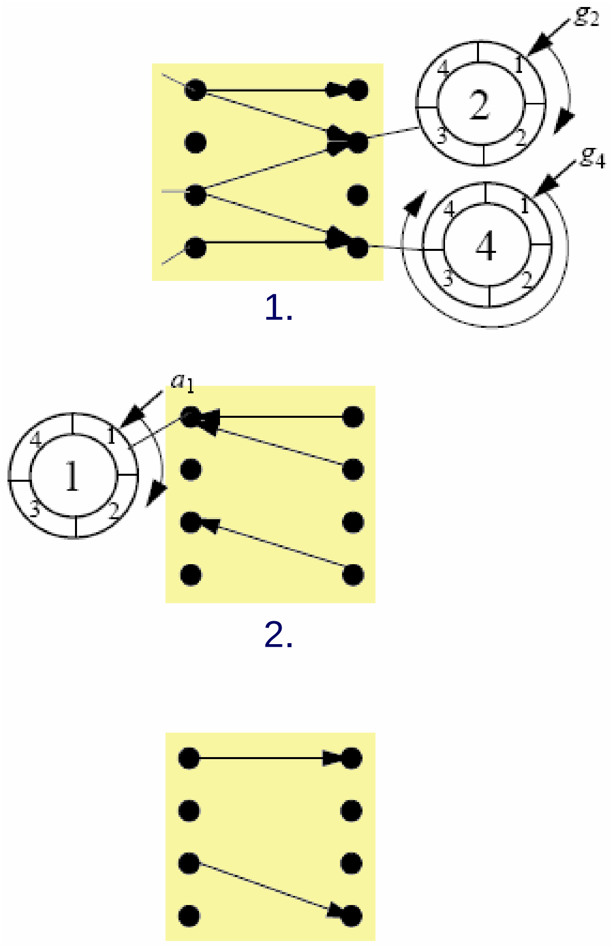
Practical algorithms: randomized versions



- In an input-queued switch with *N* ports, MaxWeight can be computed in time $O(N^3)$
- *N* may be 30 to 300, so MaxWeight is impractical

- Randomized MaxWeight
  - Each timestep pick a new schedule at random
  - Use the last timestep's schedule, or this new schedule, whichever has larger weight
- This algorithm has 100% throughput and terrible delay performance. Tweaked versions do much better.

Practical algorithms: iSLIP

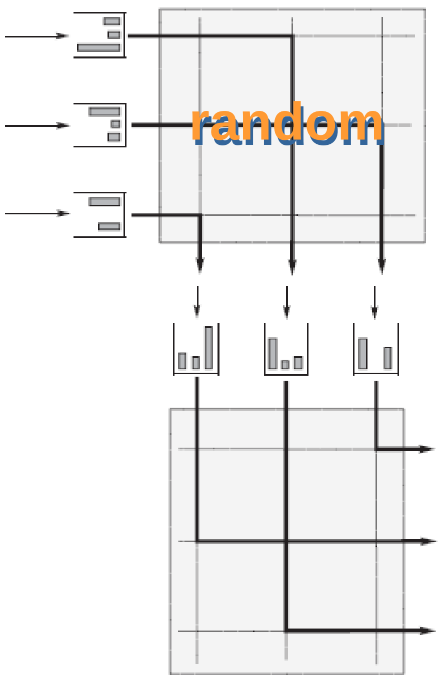


1. 2. 3.

- Each port maintains a priority ring
- For a *N*x*N* switch, run log *N* iterations of the following:
  1. Each input sends *I have packets for you* to all outputs for which there are packets waiting
  2. Each output chooses no more than one of the inputs that it heard from, giving preference to the next input in its priority ring, and replies *I want your packets*
  3. Each input chooses no more than one of the outputs that it heard from, giving preference to the next output in its priority ring, and replies *OK, let's match*
  4. Any matched input-output pairs move their priority pointers to
     (port they matched to +1) mod N

Figures from N.McKeown, 1999

Practical algorithms: the Chang method



- Suppose the traffic matrix is sub-uniform, $\lambda_{m,n} \leq 1/N$ for all m,n
  - Then we can use a static round-robin scheduling policy
- Chang's ingenious idea is to send incoming packets to a random destination, and then to switch them to the correct destination
  - the traffic matrix at each stage is sub-uniform, so a static round-robin scheduling policy works