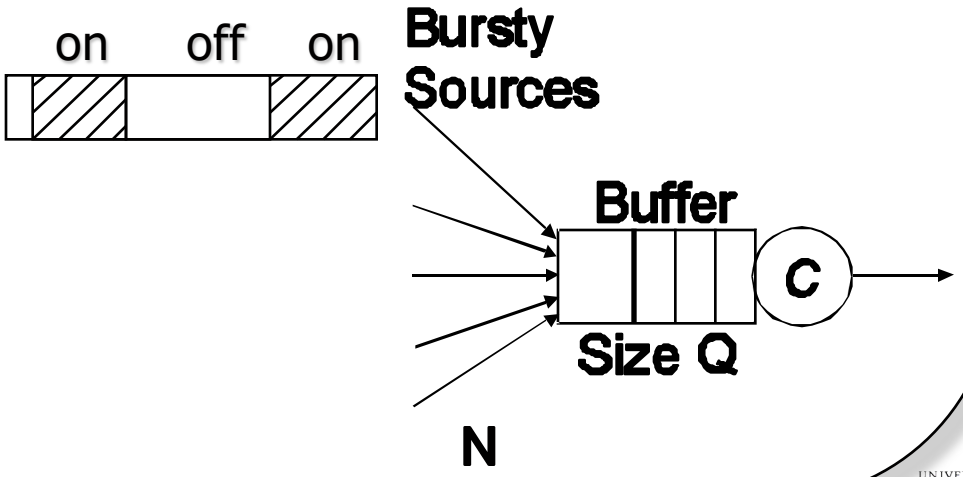
Efficient Distributed Simulation of a Communication Switch with Bursty Sources and Losses

A.S.McGough I.Mitrani

University Of Newcastle Upon Tyne

Model of a Communication Switch

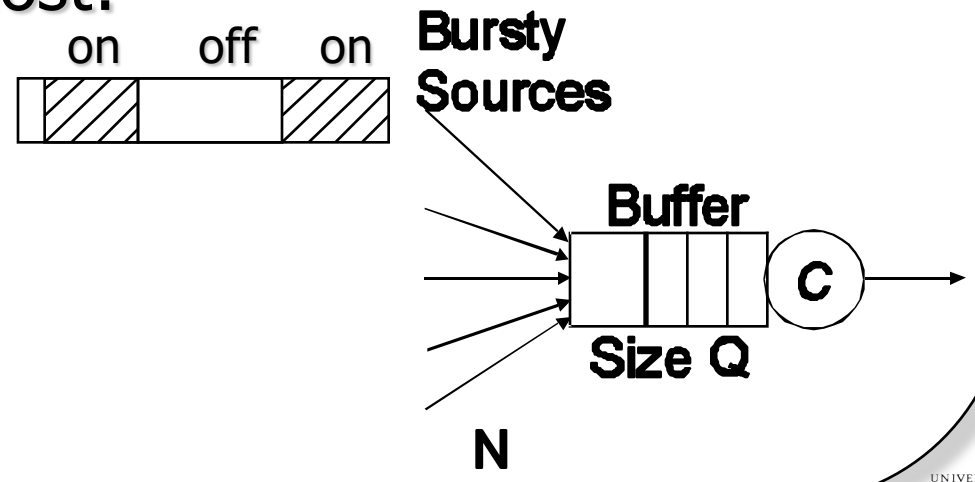
- Cells are generated by N independent bursty sources.
- There is an independent sequence of Off/On periods for each source.



- The communication switch has a finite buffer of size Q, where cells are stored in order of arrival and the switch capacity is C cells per unit time.
- Cells finding a full buffer are lost.
- Performance measure:
  - Proportion of Cells lost:

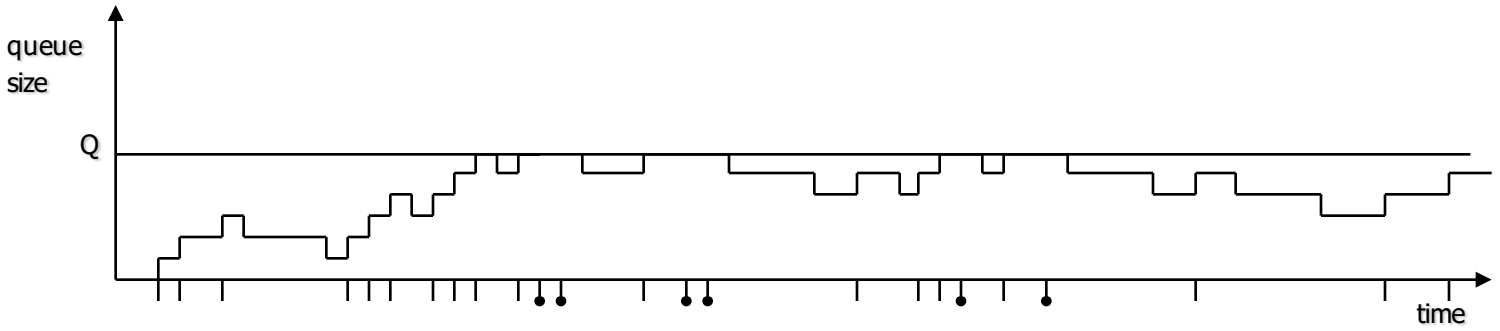
    Lost Cells /

    Total Cells.



Objective

To generate the sample path

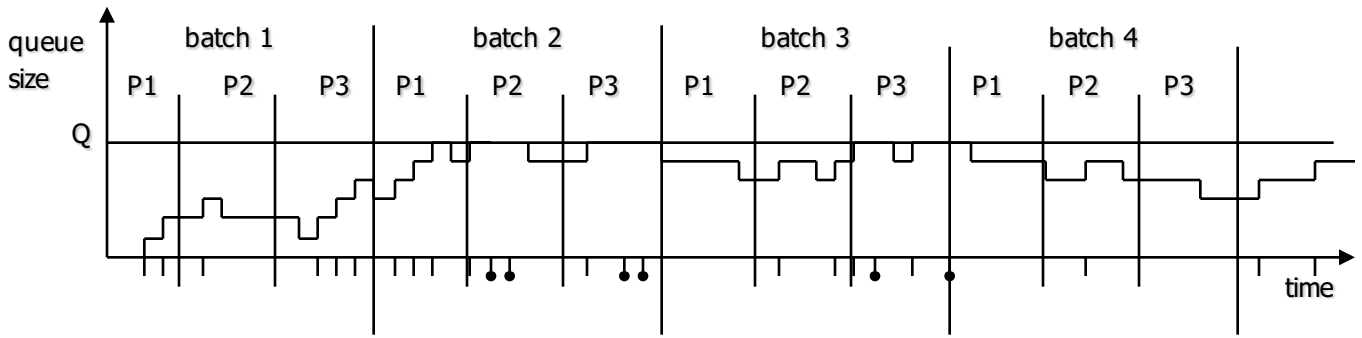


On a cluster of distributed workstations.

Problem: communication costs.

– Reduce communication costs by generating results locally.

Approach to solution



- Processor *k* can do most of the work for its partition independently
  - A little info is required from processor *k-1.*
- Recurrence equations used to generate arrival and departure times.
- Relaxation to resolve uncertainties.

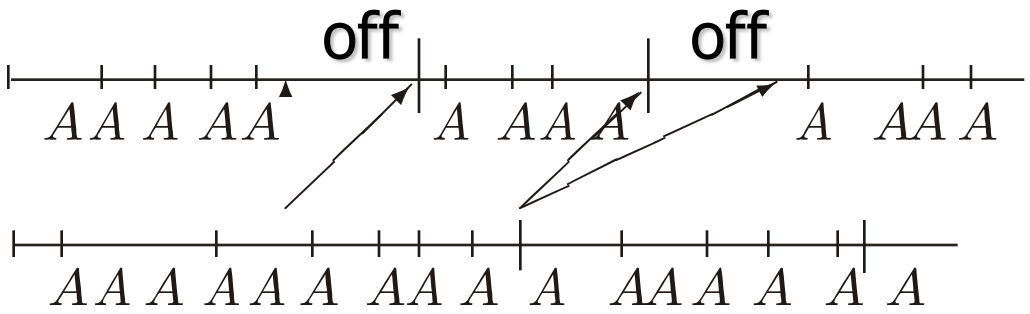
Stages of the simulation

Simulation proceeds in intervals of length B, each of which is processed in parallel by P processors.

- Generate arrivals in parallel from each stream.
  - Arrivals are generated on the processor where they will be used.
- Merge arrivals locally on each processor.
  - No need to communicate arrivals between processors.
- Mark and remove lost cells.
  - Performed by relaxation.
  - Computes buffer occupancy.

Generate arrivals in Parallel

- Generate arrivals in stream until we have filled the interval.



- Recurrence relation for arrivals:

$$A_{n+1} = A_n + \alpha_{n+1}$$

- Solve by a modified version of the parallel prefix algorithm.

(above recurrence is associative)

Generation of Arrivals

- Performed using modified Prefix computation
  - Each processor is allocated time interval.
  - Computes which on and off periods happen in this interval.
  - Compute individual arrivals during the interval.
  - May need to communicate a few that are not correctly placed.

Mark and remove lost cells

- Algorithm:
  - works with arbitrary batch sizes.
  - computes the queue size seen by each arrival.
  - only state of queue after the last cell of a batch needs to be kept for the next batch.
- Solves sets of recurrence relations in parallel, and uses relaxation.

Acceptance Algorithm

- q_n is the queue size just before cell n arrives.
- d_n is the time of the last departure before cell n arrives or A_n if $q_n = 0$.
  - Solve the recurrences:

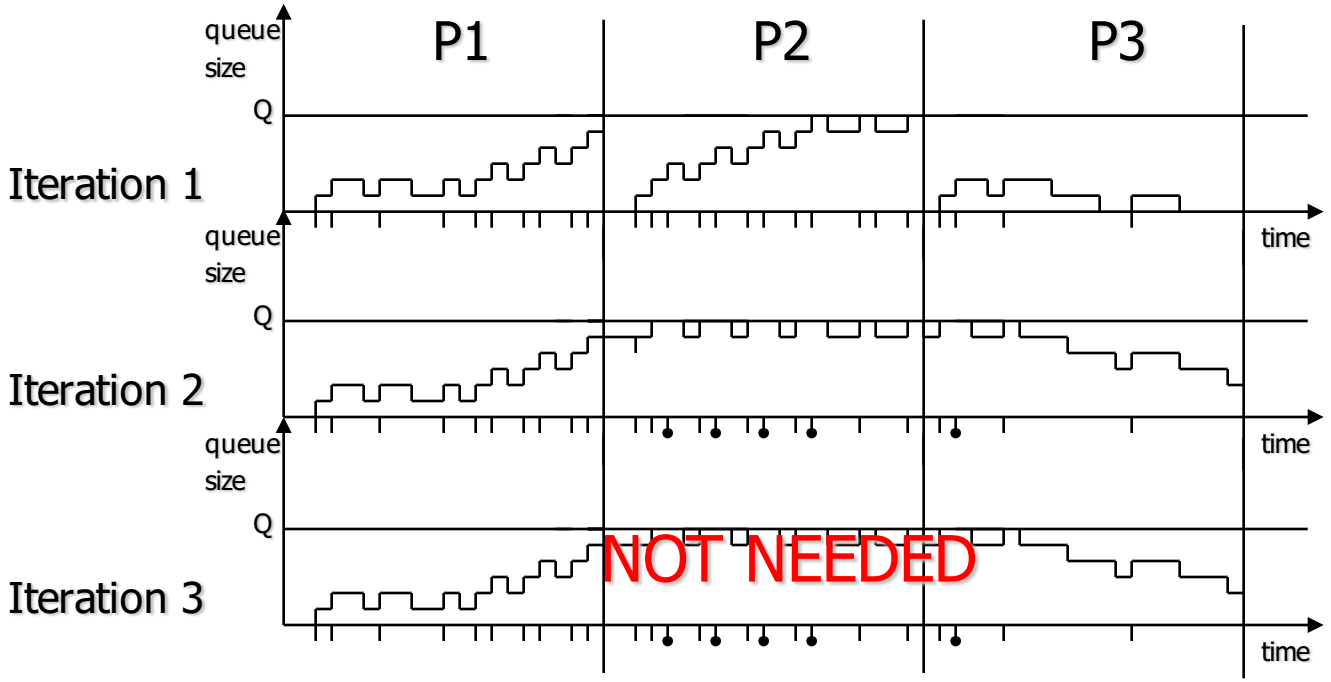
$$q_{n+1} = \max(q_n + \sigma_n - \delta_n, 0)$$

$$d_{n+1} = \begin{cases} d_n + \delta_n c & \text{if } q_{n+1} > 0 \\ A_{n+1} & \text{if } q_{n+1} = 0 \end{cases}$$ (not associative)

σ_n is 1 if cell *n* was accepted 0 otherwise.

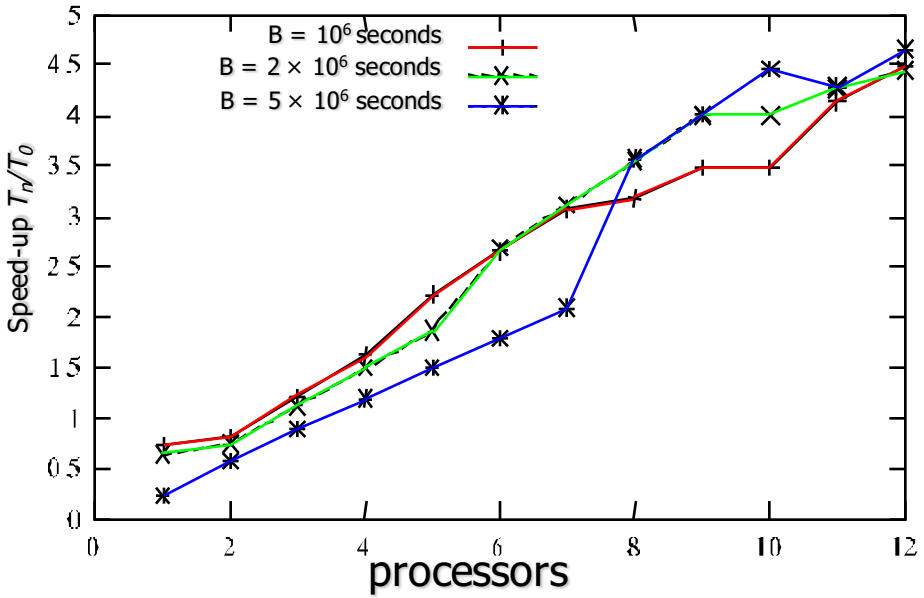
δ_n is the number of cells serviced since cell *n* arrived

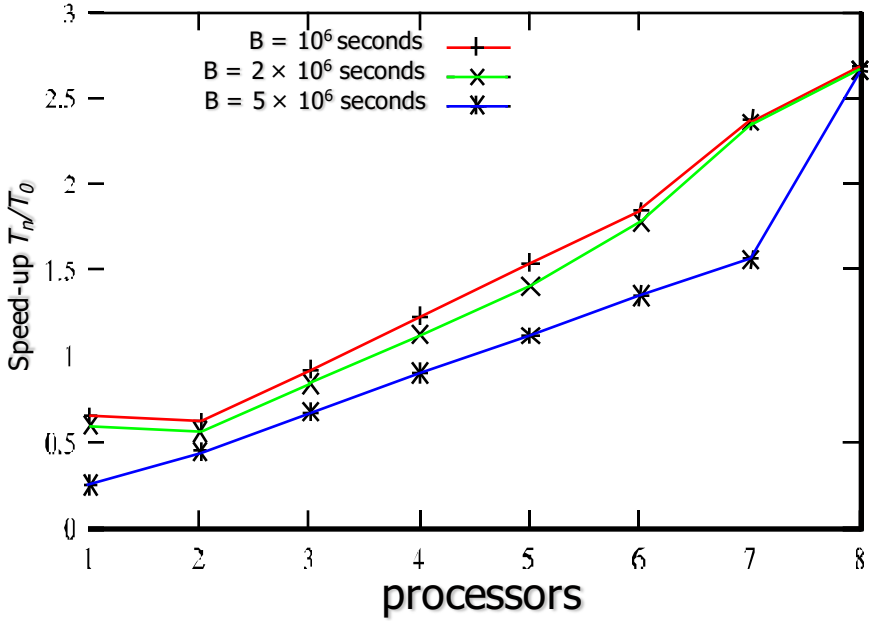
- Produce first guess at sample path by assuming start conditions.
- Use end conditions from processor *k* as start conditions for processor *k+1* on the next iteration.



Results

- Performed on a cluster of Linux workstations.
- Connected over fast Ethernet.
- Graph of 6 stream inputs
- Graph of 24 stream inputs





Conclusion

- Speed-up obtained is almost linear.

$$T \sim O(B/P)$$

- This holds even in cases where cell loss is relatively high (1%).
- All random variables can have arbitrary distributions.
- Can make this model more general.