Case study: InfiniBand CM Parameter Tuning

Mitch Gusat and Wolfgang Denzel
with IBM team

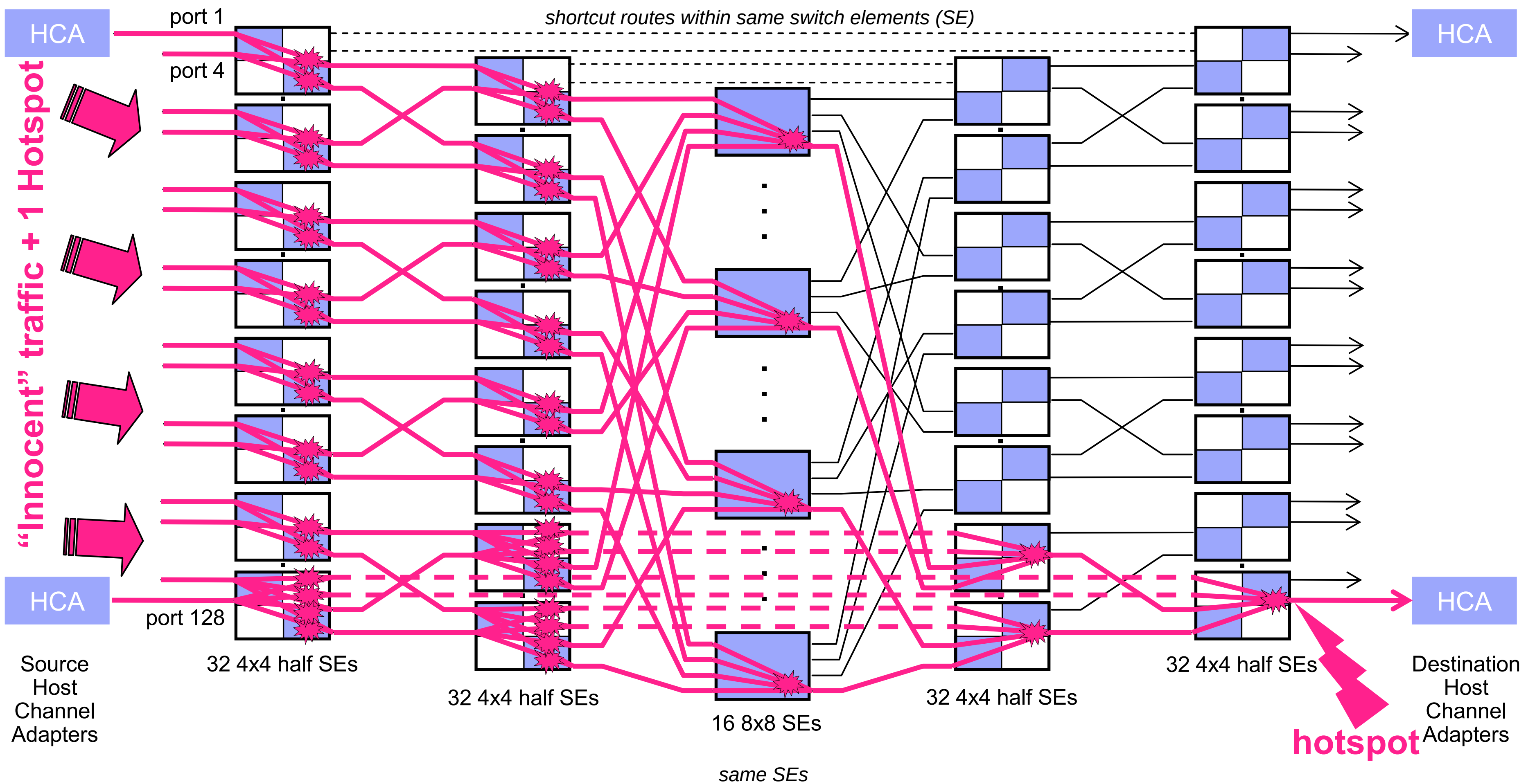
IBM ZRL

IEEE 802 Plenary Session
Dallas, Nov. 2006

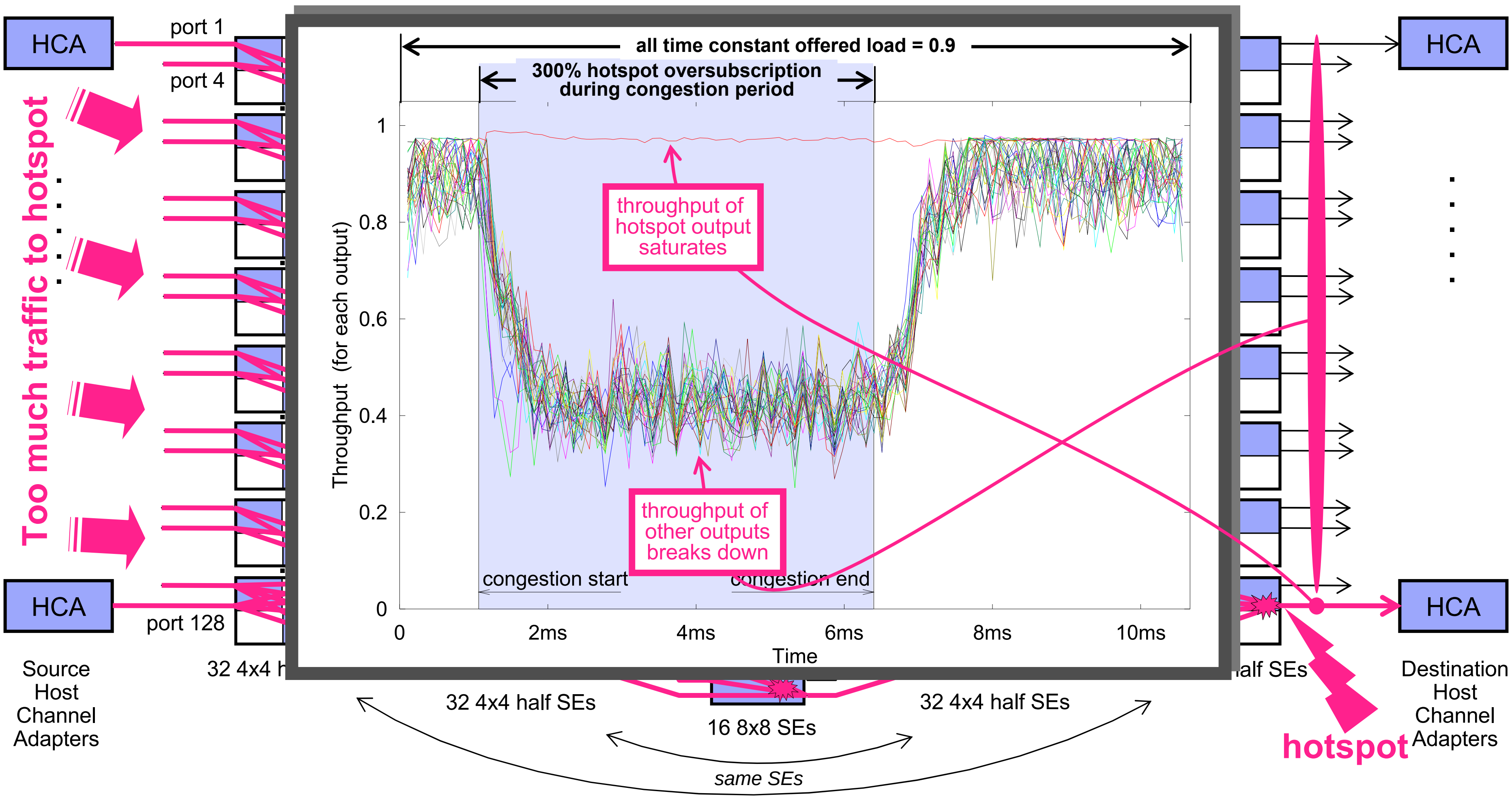
Outline

- Problem: Congestion in IBA Networks
- Solution (elements thereof): CCA
- Need: Tune the CCA parameters
- Method: ZRL Congestion Benchmarking
- Simulation results and conclusions

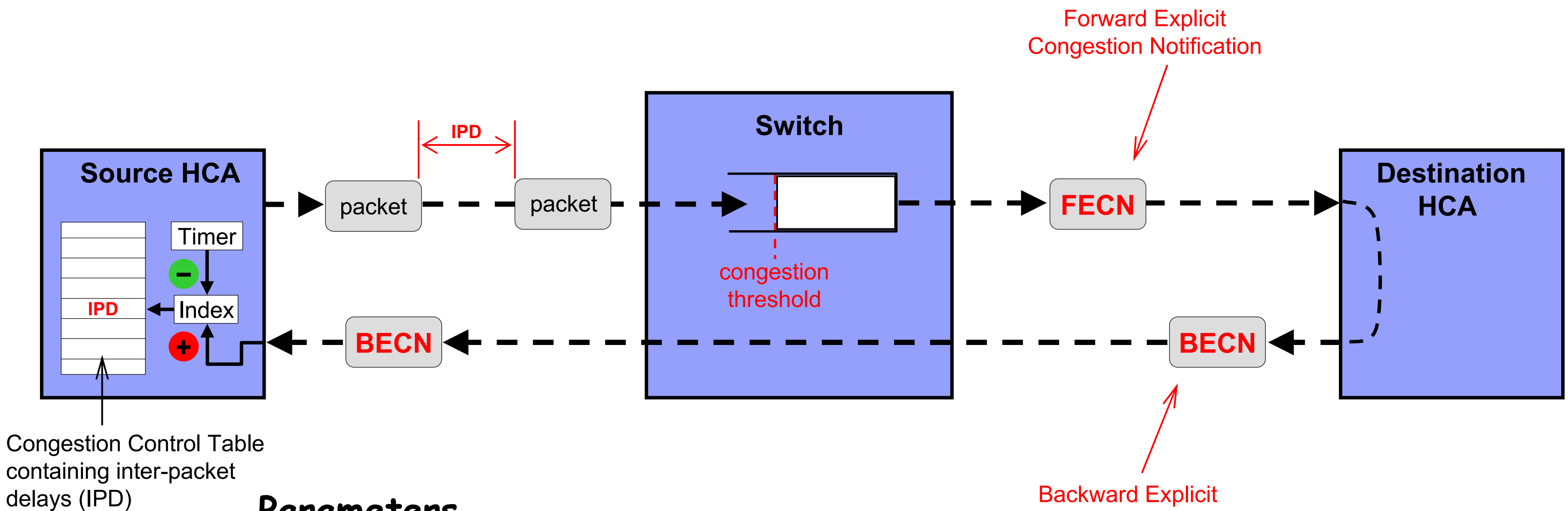
Problem: Hotspot Congestion in Lossless ICTNs => Saturation Trees



Effect: Global collapse of throughput caused by single hotspot



Solution (elements thereof): IBA Congestion Control Annex (CCA)



Parameters

- Switch queue threshold ($\sim Q_{eq}$): *sw_th*
  - congestion detection and FECN marking
- IPD table size = 64-256 entries
  - dynamic range of per flow rate control
- highest IPD entry: *max_ipd*
  - largest inter-packet gap = $1/R_{min}$
- IPD index increment: ***ipd_idx_incr***
  - step on each new BECN => rate control granularity
- IPD recovery timer: *rec_time*
  - timeout of the autonomic rate increase timer

Questions

1. Does it work?

2. Tuning: What parameter settings for which cases?

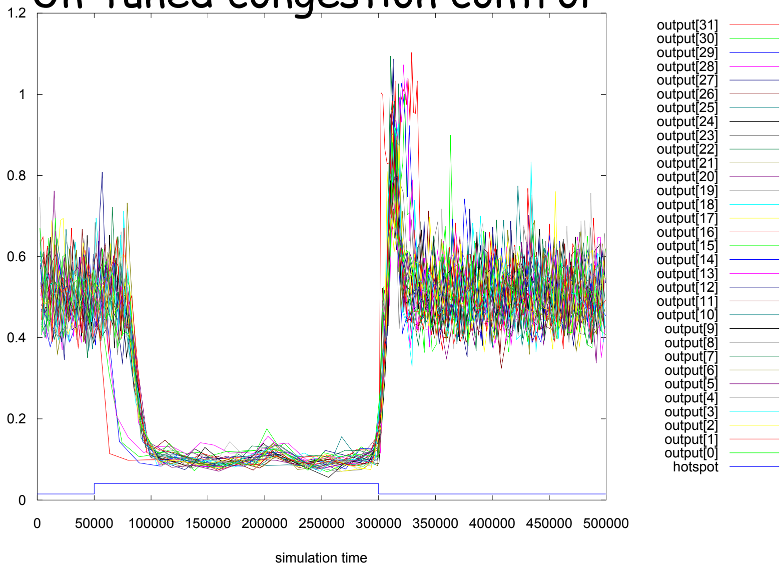
   => exploration of a combinatorial sub-space

3. Where are the limits of its operation?

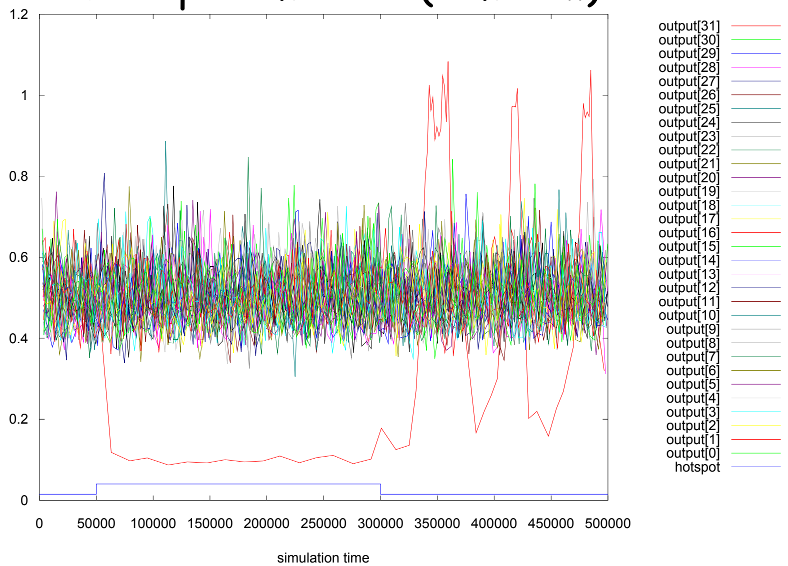
Does it work?

- Qualified "yes" => needs tuning
  - easy for small fabrics w/ simple traffic, hard for others...
- Param *tuning* required per (1) fabric architecture and (2) traffic pattern
- Fragile stability (stiffness): CCA sensitivity to traffic and params...

Un-tuned congestion control



Tuned parameters (almost..)



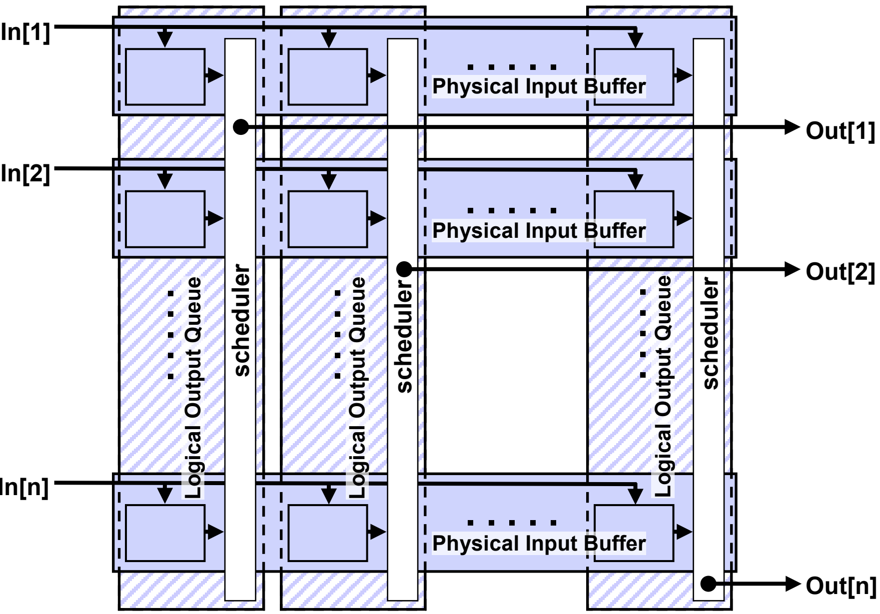
Early Observations

- With min. pkt size under high background load, ECN traffic can lead to instability
- A hotspot relay destination receiving permanent short-packet traffic may be overloaded with F/BECN signaling
- BECN and ACK packets can be caught in reverse congestion
- Reaction delay improvable by direct BCN vs. the CCA mechanism
- CCA operation is sensitive to parameters
  - Oscillations observed... seem to be controllable
- Parameter settings strongly depend on fabric and traffic!

Simulation Model

- Switching network
  - 4x4..12x12 input-buffered CIOQ switches
  - Credit-based flow control
  - Multistage bidirectional fat-tree network
  - Source routing: Static or dynamic (shortest path)
- Edge adapter (HCA) operation
  - Queue pairs, credit-controlled round-robin scheduling with rate ctrl. triggered by CM
  - Destination will ACK (64B pkt) after successful reception of user packets (MTUs)
  - Return of short BECN packets if FECN bit is set
- CM mechanism
  - **Switch action**: Output buffer threshold with hysteresis -> FECN bit set in packets leaving switch module and BECN packets returned from destination HCA
  - **SRC HCA action**: Rate control according to CCA.

Baseline Switch: Xbar-based CIOQ with Input Buffering

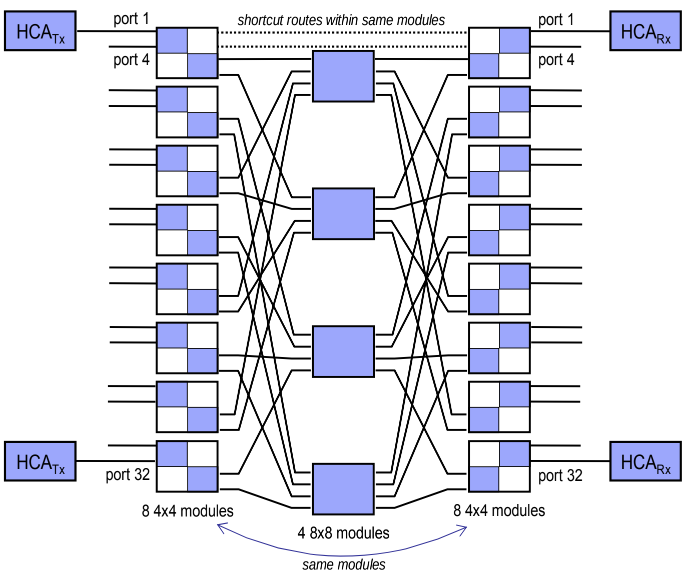


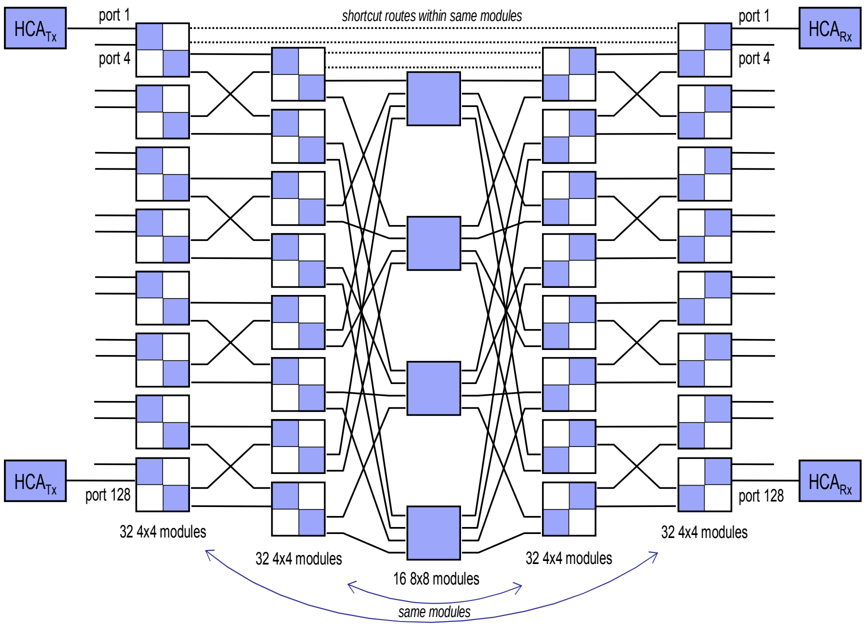
Switch Architecture
IN buffer = 72KB
Output service: RR

Switch element (SE) described in Omnet++

```
#include <omnetpp.h>
#include <packetdefinition.h>
class Switch : public cSimpleModule
{
  Module_Class_Members( Switch, cSimpleModule, 0 )
  < some parameter and variable definitions …….. >
  virtual void initialize();
  virtual void handleMessage( cMessage *msg );
  virtual void finish();
}                                    Typical 3-method structure
Define_Module( Switch );
void Switch::initialize()
{ 1. performed before simulation run
  < some initializations, allocation of queue arrays,
     schedule first SCHEDULE_EVENT, …….. >
void Switch::handleMessage(cMessage *msg)
{ 2. handling of received messages during simulation run
  switch( msg->kind() )
  {
  case PACKET:           < en-queue received packet …….. >
  case SCHEDULE_EVENT:   < de-queue and send packet, return credit,
                           update scheduler pointer, schedule next
                           SCHEDULE_EVENT, ….. >
  case CREDIT:           < return credit to credit pool …….. >
  }
}
void Switch::finish()
{ 3. performed after simulation run
}
```

Baseline Fabric: MIN Topology => Bidir Fat Tree





- 2-level / 3-stage bidir MIN
- Simulated: 8 – 32 nodes
- Time per run: 2-30 min.

- 3-level / 5-stage bidir MIN
- Simulated: 128 – 432 nodes
- Time per run: 20-500 min.

Traffic: ZRL Congestion Benchmark

- Source nodes generate one or more hotspots according to matrix $[\boldsymbol{\lambda}_{ij\_hot}]: t_{p\text{->}q} = \alpha_{k\_hot} [\boldsymbol{\lambda}_{ij\_hot}]: t_{p\text{->}q}$, $[\boldsymbol{\lambda}_{ij\_hot}]$ is specified per each case below

1. Congestion **type**: IN- or OUTput-generated

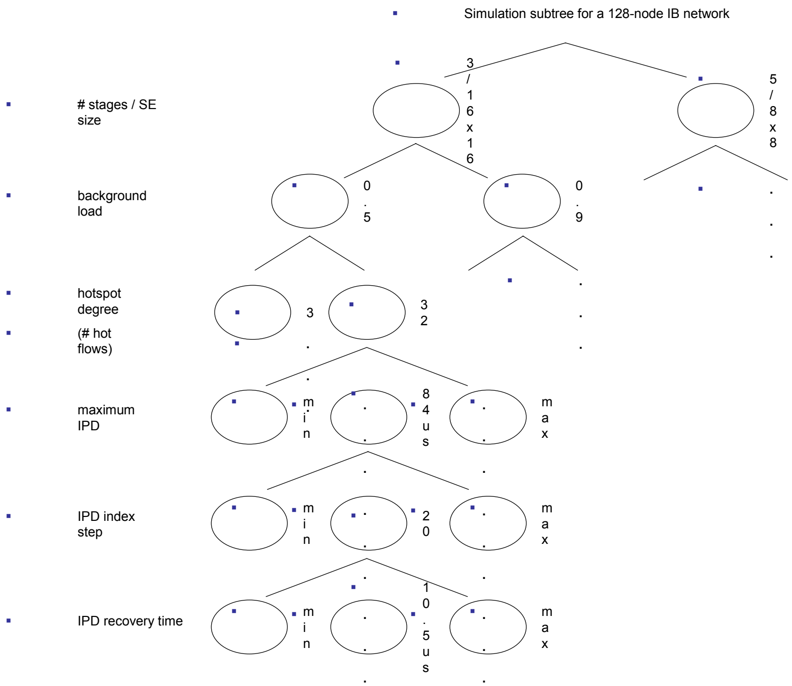
2. Hotspot **severity**: HSV = $\boldsymbol{\lambda}_{aggr} / \boldsymbol{\mu}_{HS}$, $\lambda_{aggr} = \sum \lambda_i$ at hotspotted output, μ_{HS} = service rate of the HS
   - *Mild* HSV ~ 1 .. 2
   - *Moderate* HSV = 3 .. 7
   - *Severe* HSV >> 10.

3. Hotspot **degree**: HSD is the fan-in of congestive tree at the measured hotspot
   - *Small* HSD <10% (of all sources inject hot traffic)
   - *Medium* HSD 20..60%
   - *Large* HSD >90%.

Sensitivity Analysis: Combinatorial Tree

- 1 sim. point: ~ 1hr
- Exhaustive coverage: ~ 78 yrs.
    - output: ~ 23 PB of data…

      not feasible, therefore…
- Prune the tree by analytical pre-selection
    - Network:
        - 32-node 3-stage/128-node 5-stage
    - Traffic:
        - single HSV = moderate congestion
        - two HSD = 3 and 32 flows



Via analysis and validated by simulation, rank CCA params by sensitivity:

1. ipd_indx_incr
2. rec_time
3. ipd_max

ca 800 cases were analyzed

Sample Results: "Input-generated" Congestion

Selection: 4 'corner' cases

Offered load

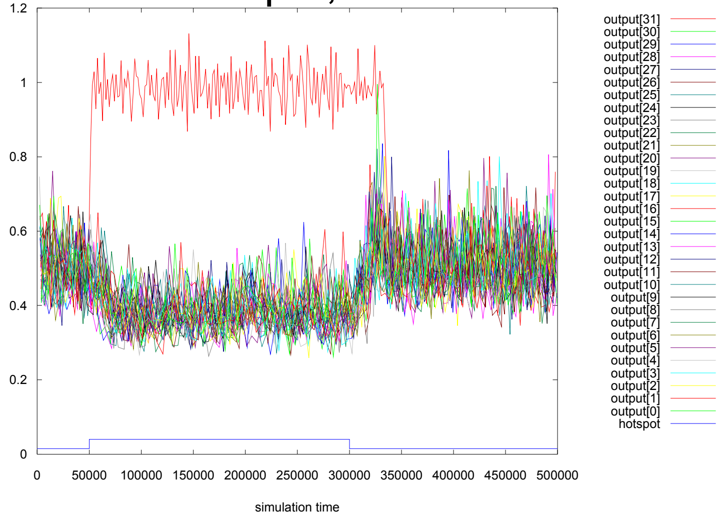
- pkt size = 2KB (all user traffic MTU)
- negative-exponential interarrival times
- background traffic destinations uniformly distributed, load:
  - a) 50%
  - b) 90%

2 congestion patterns, both with HSV ~ 3

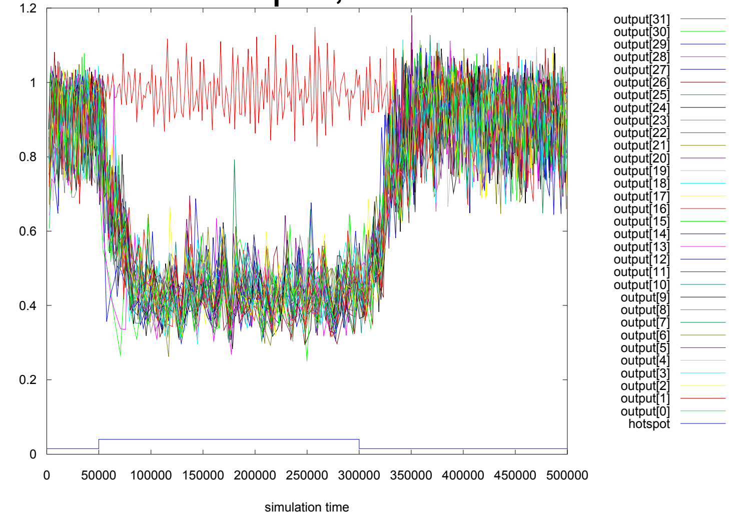
- Small HSD:
  - During hotspot period, 3 sources send 100% of their load to DST #31
- Large HSD:
  - During the hotspot all 32 sources send 10% of their load to DST #31

Four corner cases: Un-tuned CM ~ No CM

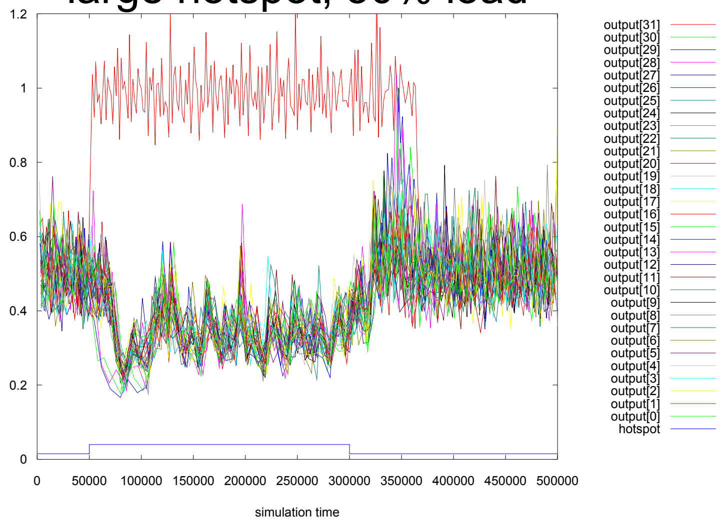
small hotspot, 50% load



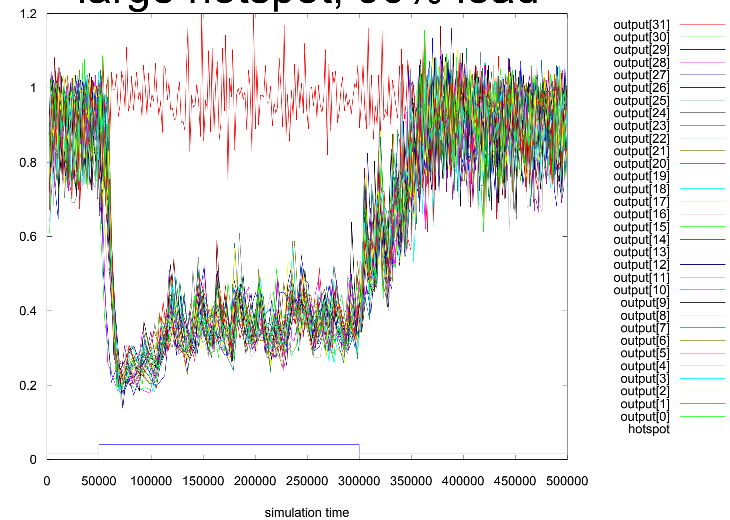
small hotspot, 90% load



large hotspot, 50% load



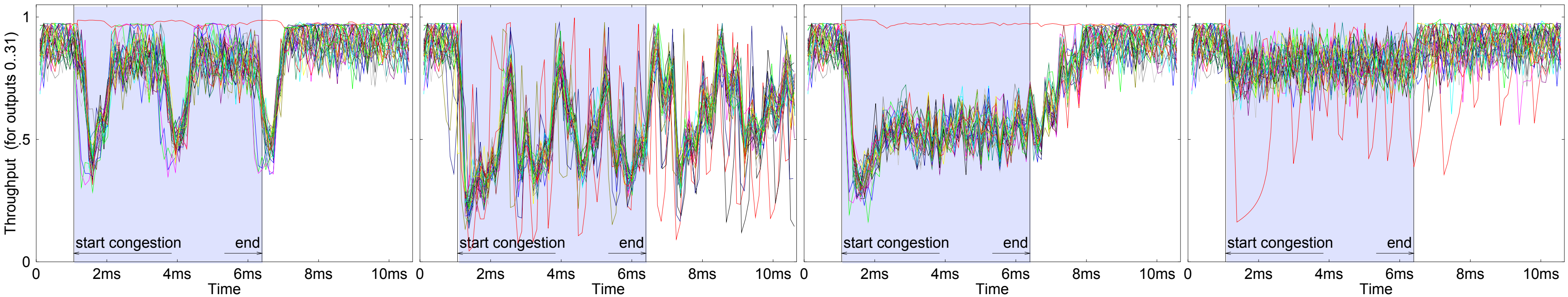
large hotspot, 90% load



Focus: 4th corner case

Corner case:	4		
Load:	High (0.9)		
Hotspot degree:	Large (32)		
Out-of-range parameters			
IPD index step too low (=2)	IPD index step too high (=40)	IPD recovery timer too fast (=2.6us)	IPD recovery timer too slow (=84us)

Throughput: with Congestion Control:



oscillations | oscillations | breakdown | hot flow suffering

32x32 3-stage (2-level) fat-tree

Common parameters: ***Packet size*: 2 KB** ***Hotspot severity*: 300%**

With marginal settings of *ipd_idx_incr* and *rec_time*

Tuning for IG congestion in 32 to 128-node fabrics

32-node

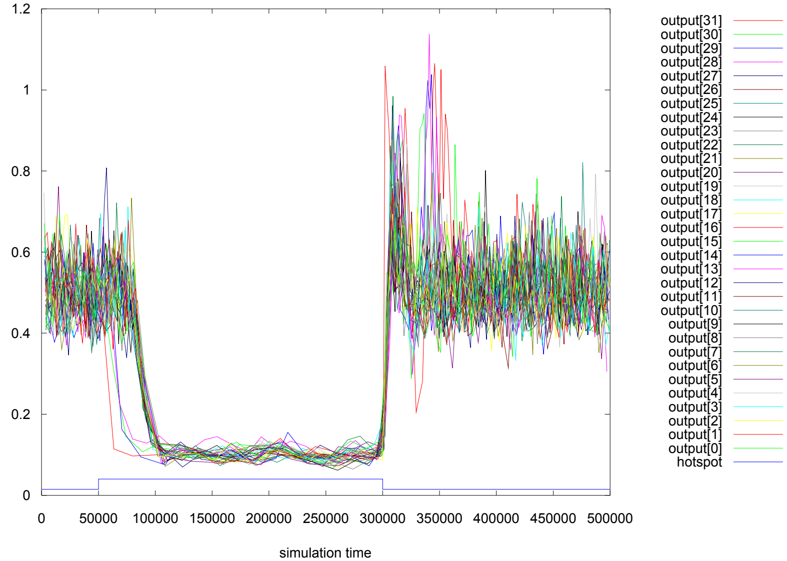
- *sw_th* = 90% of switch input buffer size
- *max_ipd* = 1000 (= 21μs)
  - from fairness under worst case => IPD sufficient for all other N-1 sources flows to access a share (1 pkt) of the HS bandwidth
- IPD table index increment *ipd_idx_incr* = 5 (sweetspot)
- IPD recovery time *rec_time* = 500 (= 10.5μs)

128-node

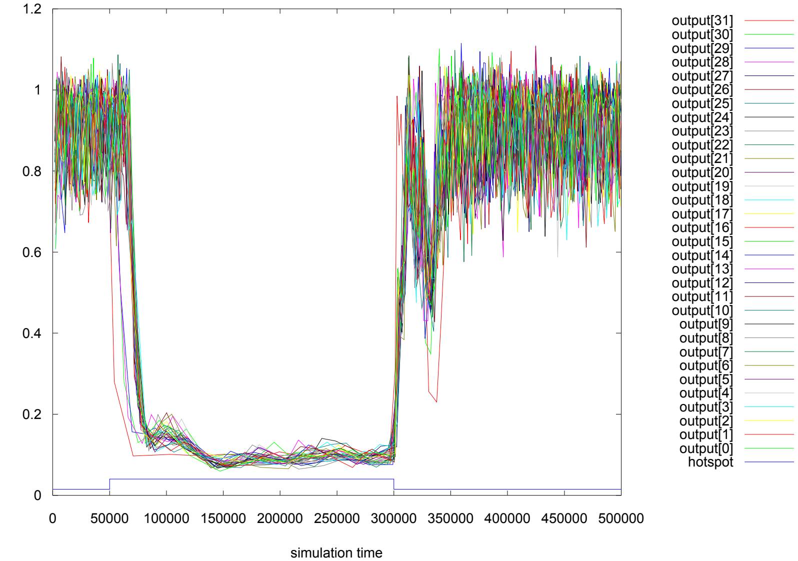
- *max_ipd* and *ipd_idx_incr* ~ 4x larger than above
  - linear scalability confirmed by simulations

Output-generated congestion with CM tuned for IG

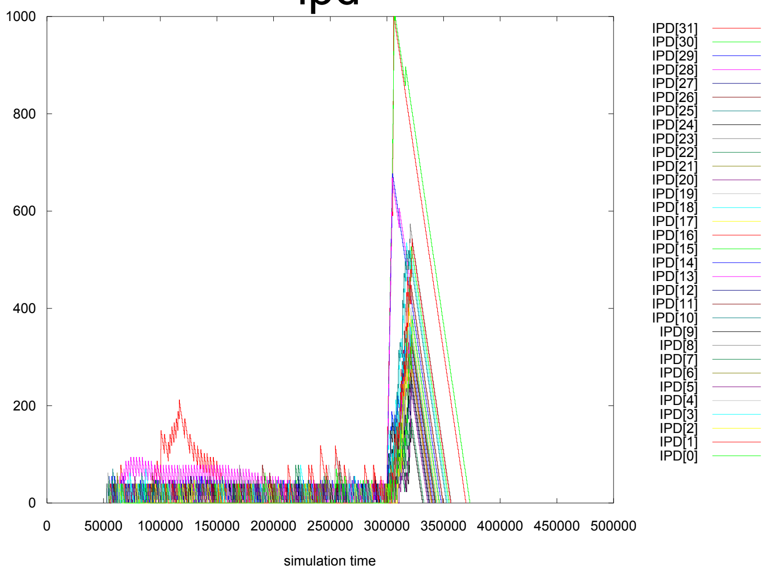
50% load



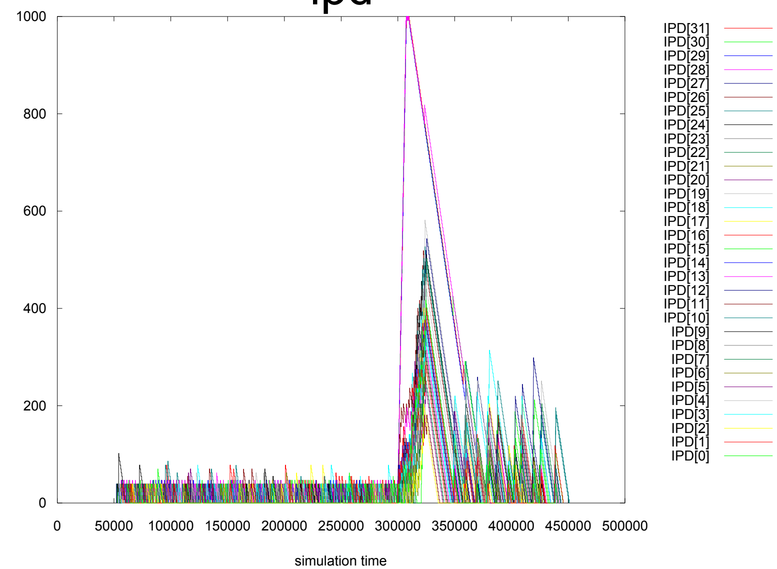
90% load



ipd

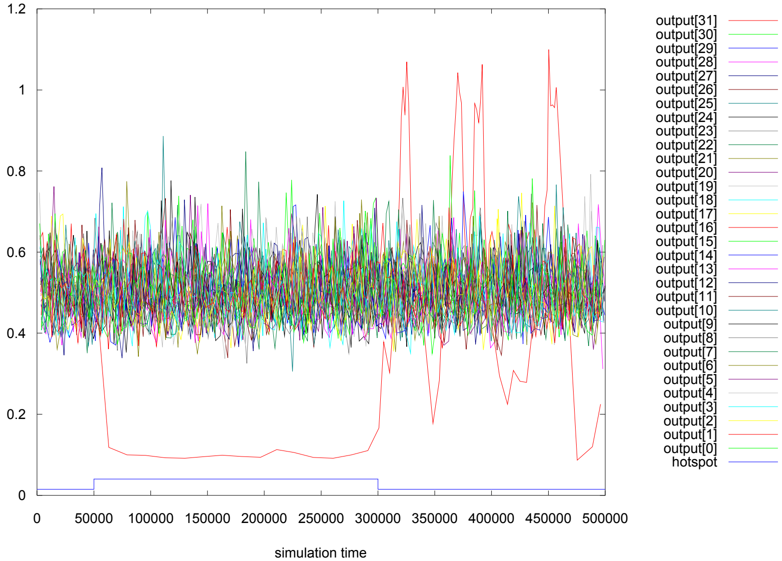


ipd

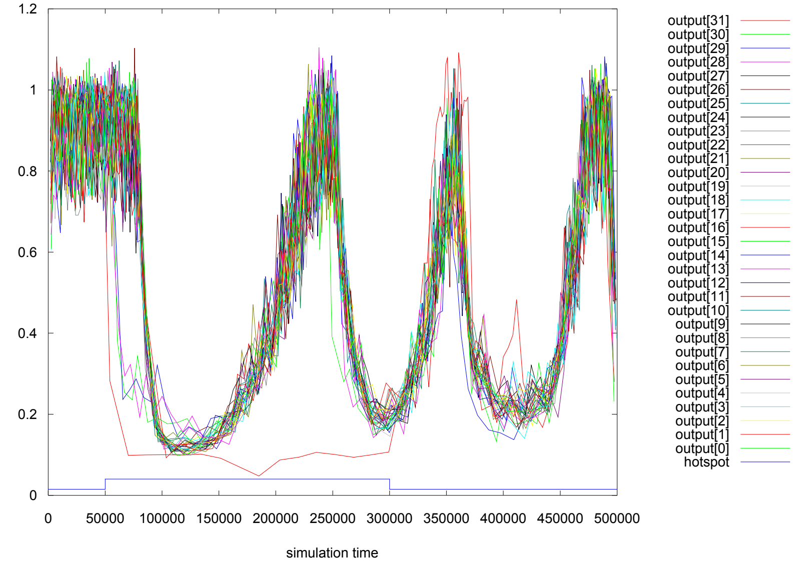


OG w/ “Big Bang” control

50% load



90% load



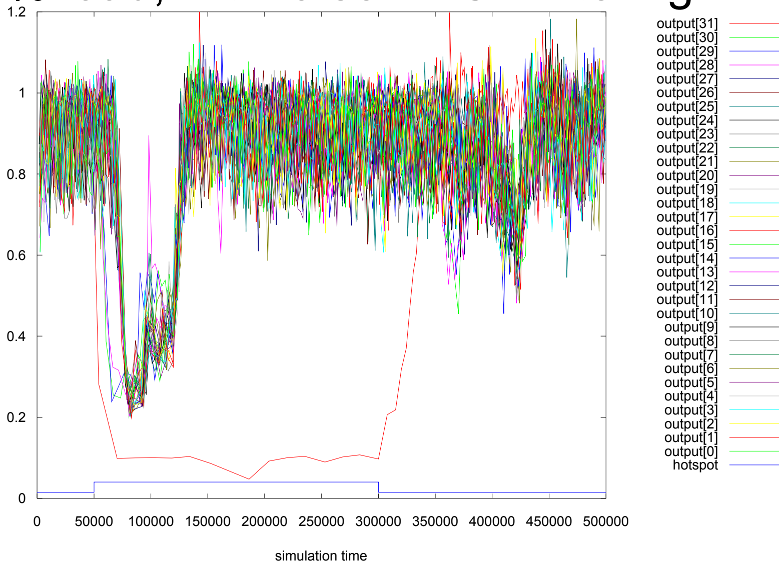
Parameters for OG:

1. ipd table filing: multiplicative (vs. linear in the IG)
2. max_ipd = 10000 (= 210us)
3. recovery time: rec_time = 10000 (= 210us)
4. ipd increase: ipd_idx_incr[i] += 127

- ipd decrease: ipd_idx_incr[i] -= 1

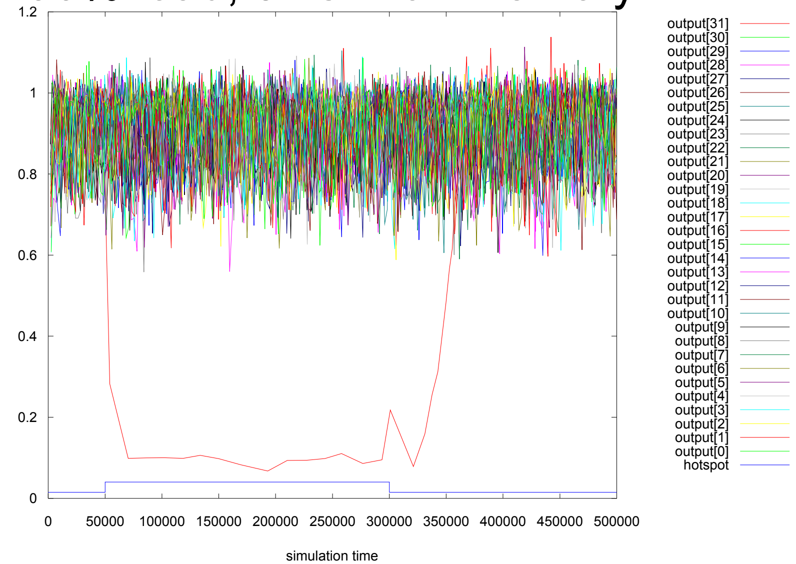
OG w/ “Big Bang” control and enhancements

90% load, with false BECN filtering



ipd

90% load, 3x switch memory



ipd

- With
  - above enhancements,
  - different (from IG congestion) control algorithm, and,
  - different table values,
- a (marginally) stable solution to OG congestion is feasible.

Tuning Guidelines for IG Congestion

- N = network size, i.e., number of ports
- HSD = maximum hot spot degree
  - Worst case = N, but can be less if system partitioning known.
  - Absolute time (μsec) values are w/ reference to our model's RTT of 2-20 μsec. (w/o congestion)

Item	**Value**
IPD Table size	128
Switch threshold for congestion detection	90% of queue capacity, with hysteresis
IPD table index increment	Min(1/6•N, 1/2•HSD)
Max IPD value	2/3 HSD μsec.
Recovery timer	10 μsec.

Limitations

- Stability limits…?
  - Large hotspot degree (128 sources send 5% to hotspot) with 50% background load has no solution in the param space
- Input- and Output-generate congestions require two distinct control loops (table values and rate control algorithms)
  - Corollary: Any mixture of IG and OG must be controlled by the dominant case, although the solution might not be stable.
- Why?
- Under-sampling
  - Generally not enough true BECNs for convergence
  - Next problem: false BECNs

» That's all.

Contributors

N. Ni†, G. Pfister†, C. Minkenberg*, T. Engbersen*, D. Craddock§, W. Rooney§, R. Luijten* and T. Schlipf‡

** IBM Research GmbH, Rüschlikon, Switzerland;*
§ IBM Systems and Technology Group, Poughkeepsie, NY USA;
† IBM Systems and Technology Group, Austin, TX USA;
‡ IBM Systems and Technology Group Boeblingen, Germany.

Contact: mig@zurich.ibm.com