Achieving High Performance with Darwin's Fairness Algorithm

Ed Knightly
Rice University
http://www.ece.rice.edu/~knightly

Contributers:

L. Balzano, V. Gambiroza, Y. Liu, and P. Yuan (Rice)
S. Sheafor (Vitesse Semiconductor)
H. Zhang (Turin Networks)

Outline

- RIAS-Fair
  - Reference model for packet rings with spatial reuse
  - Define now or else "TCP/GigE is fair"
  - Darwin approximates RIAS-Fair in most cases
  - Does not require changes to RPR-fa
- Simulation benchmark scenarios
  - Important cases for validation, performance analysis, and comparison with GigE
- Potential for enhanced Darwin-compatible fairness algorithms
- Simulation results

Reference Model

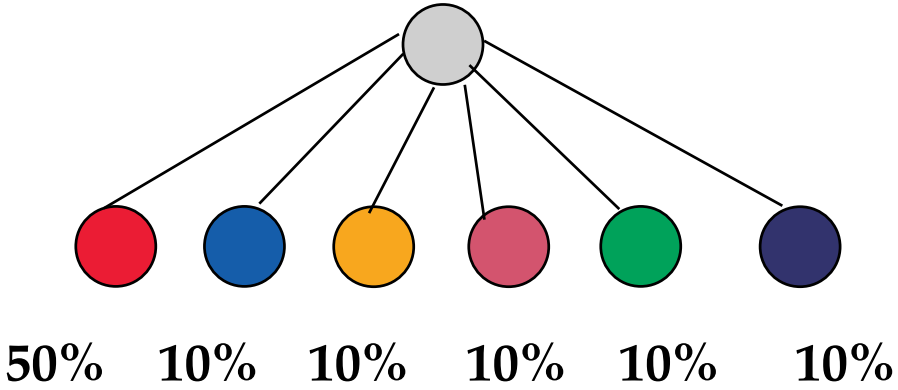
- What is a reference model?
  - Idealized goal
- Why do we need one?
  - Clear target for algorithm design
  - Quantify tradeoffs (simpler hardware design vs. deviation from ref. model)
  - More productive to compare different algorithms to a reference model than to each other
  - Preemptive strike against GigE unfairness

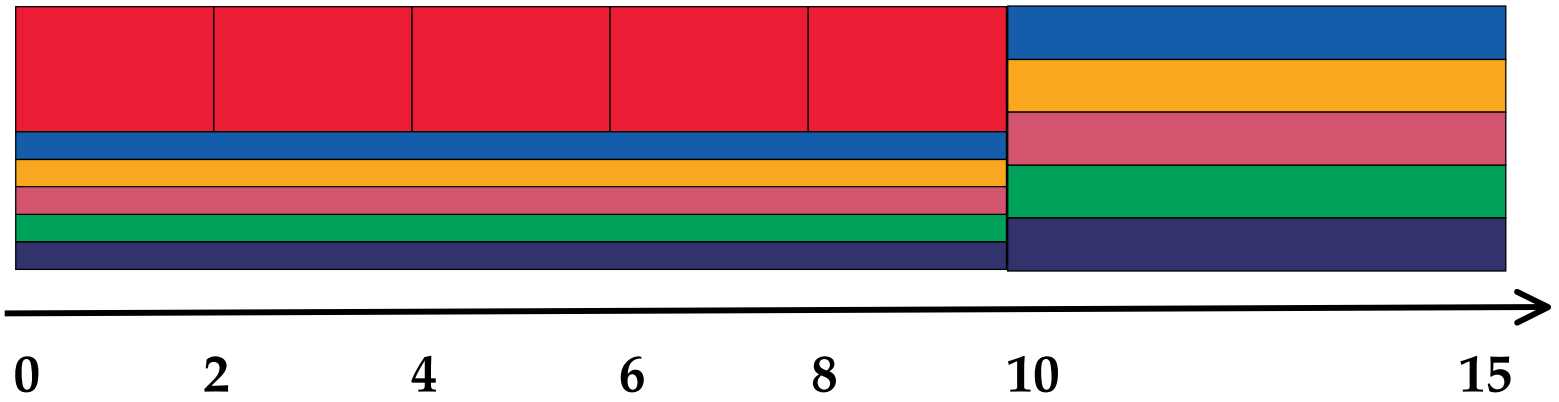
Successful example: fair queueing

Success Story: GPS and Fair Queueing

- Generalized Processor Sharing (GPS)
  - When would packets be serviced if packet service is perfectly fair and "fluid"?

- Red session has packets backlogged between time 0 and 10
- Other sessions have packets continuously backlogged





Success Story for Reference Model

Of course, GPS is unimplementable

- Practical Algorithms: DRR, WRR, SRR, …
  - Packets vs. fluid
  - Avoid "virtual time" calculation
  - Avoid sorts
  - …
- Always characterize deviation from GPS (fairness index)
- Easy to compare algorithms, make good engineering choices

A Reference Model for RPR

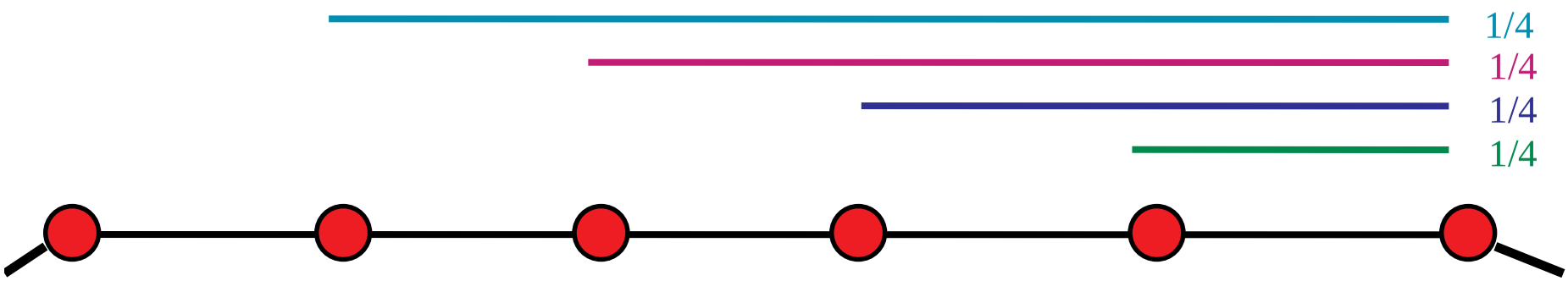
- Given all instantaneous demand rates (fluid offered traffic)
- Compute instantaneous rate-limiter values such that
  - All "flows" obtain a fair bandwidth share
  - Spatial reuse (and utilization) is maximized
  - There is no queueing or loss on the ring
- Even under idealized settings the reference model is not obvious
  - How to define a "flow"? How to fairly employ spatial reuse? …

RIAS-Fair
Ring Ingress Aggregated with Spatial Reuse

An operational definition

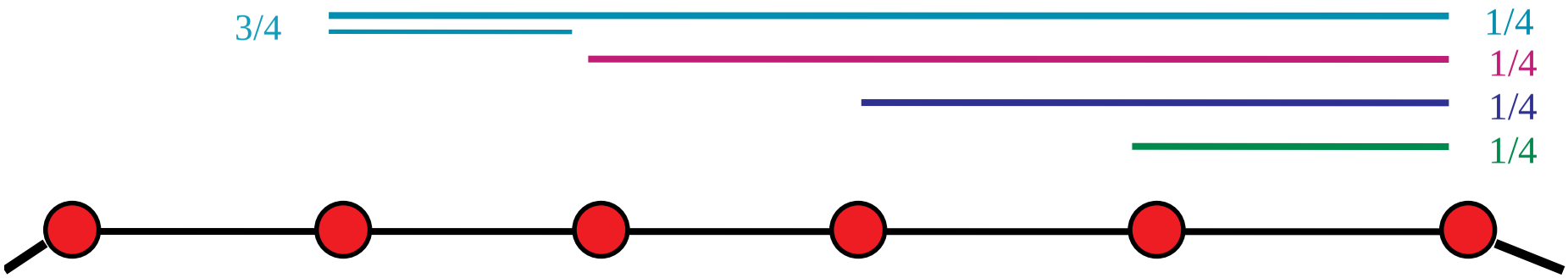
1. Fairly allocate bandwidth on each link independently according to an ingress granularity (IA traffic)
2. Refine bandwidth allocation for each IA flow according to its egress point
3. Reclaim unused bandwidth fairly by iterating

Illustration of RIAS Fair (1/3)



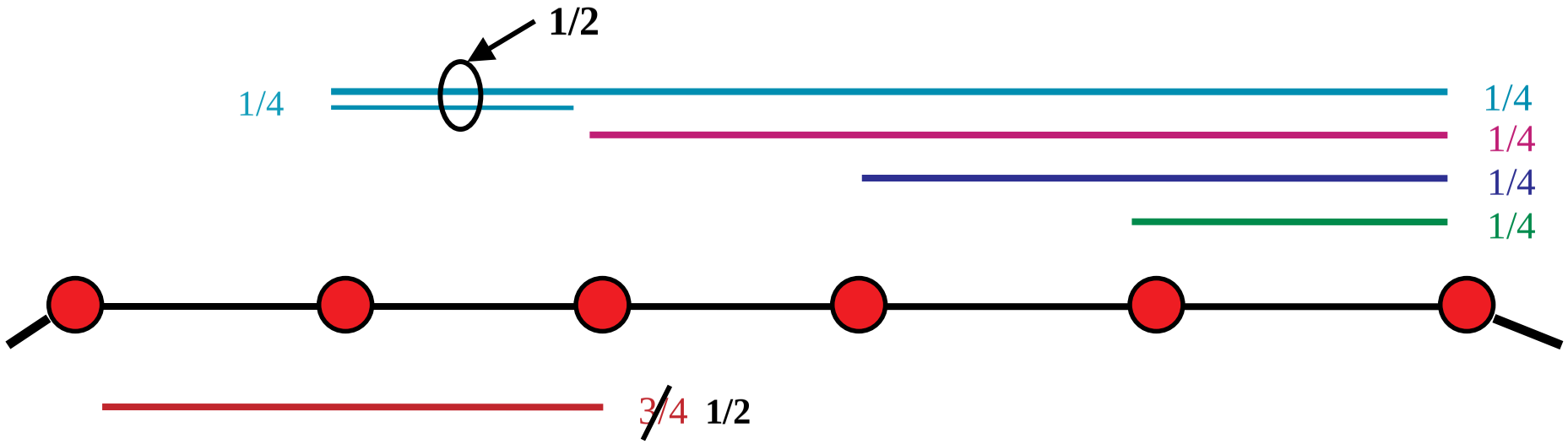
- Parking Lot
  - 4 flows each receive rate ¼

Illustration of RIAS Fair (2/3)



- Parallel Parking Lot
  - Each flow receives rate ¼ on downstream link
  - Left 1-hop flow fully reclaims excess bandwidth (RIAS)

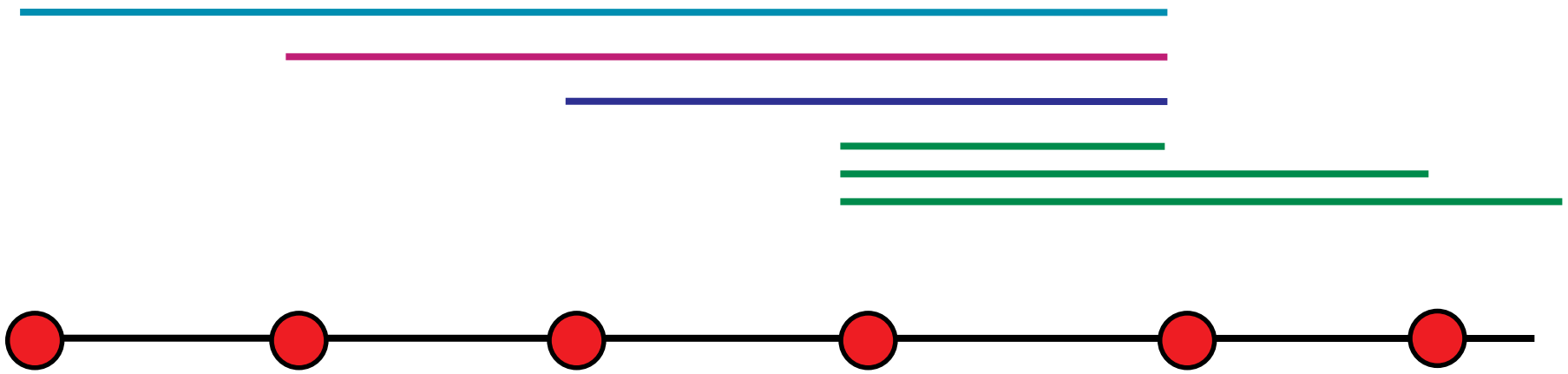
Illustration of RIAS Fair (3/3)



- Upstream Parallel Parking Lot
  - Key points:
    - Flow granularity for fairness
    - Spatial reuse

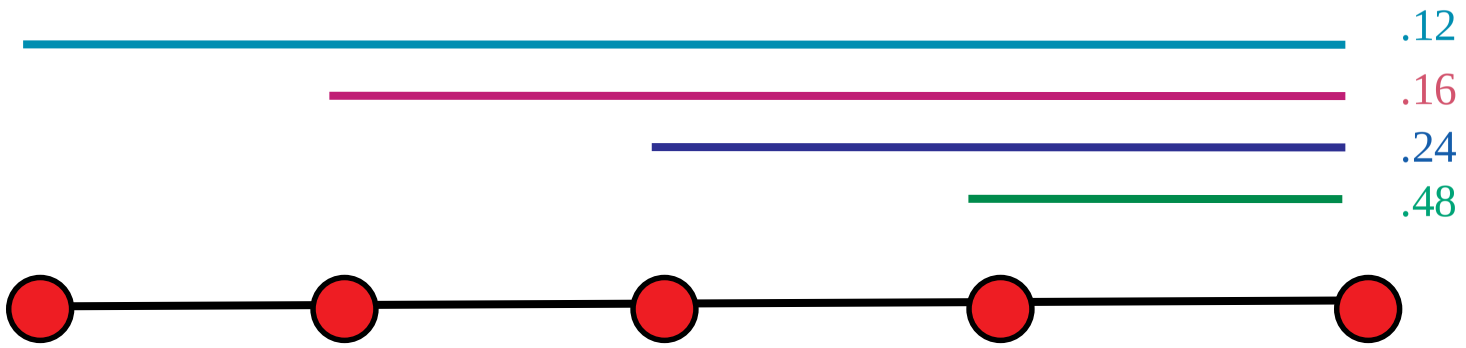
Unsuitable Fairness Model I: Ingress-Egress Flow Granularity

- Fairness with Ingress-Egress flow granularity
  - Incorrectly rewards nodes for spreading out traffic to many destinations vs. all to hub node
  - Wrong flow granularity counts 6 flows and gives rate 1/6
  - (RIAS-fair: all green flows together get ¼)



Unsuitable Fairness Model II: Proportional Fair

- Proportional fairness
  - Penalizes flows farther away from the hub
  - TCP/GigE will naturally achieve this in the parking lot
  - Service dependent on topology and traffic matrix
  - Important for TCP in the Internet (rate decreases with RTT)



- Variants of all of these have been discussed and proposed in the IEEE 802.17 meetings

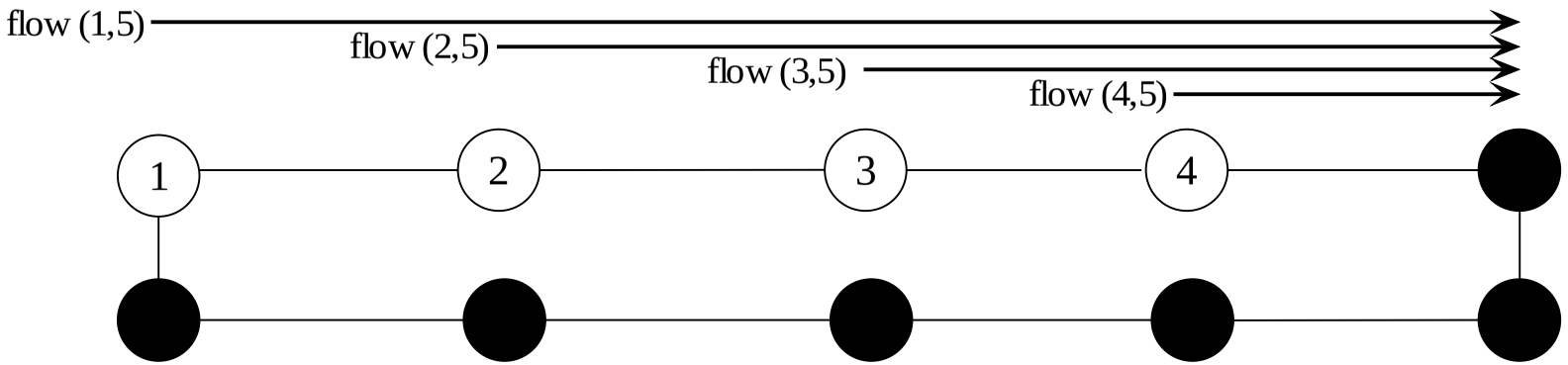
What Have We Achieved with RIAS?

- For any scenario of input traffic rates, can compute the targeted bandwidth allocations (rate limiter values) for RPR
- Algorithm's performance is evaluated on
  - Ability to converge to RIAS-fair rates
  - Time to converge to RIAS-fair rates
- Note
  1. Darwin converges to RIAS-fair rates in most cases. GigE does not.
  2. RIAS is not a fairness algorithm!

Scenarios for Performance Evaluation

- A scenario consists of
  - Set of flows (source-destination pairs)
  - Demand rates of flows
- Each scenario has at least one performance issue
  - Can RIAS rates be achieved in steady state?
    - If not algorithm has a "throughput loss"
  - Algorithm dynamics
    - Convergence time, oscillations, range and time-scale of oscillation, throughput loss due to oscillation

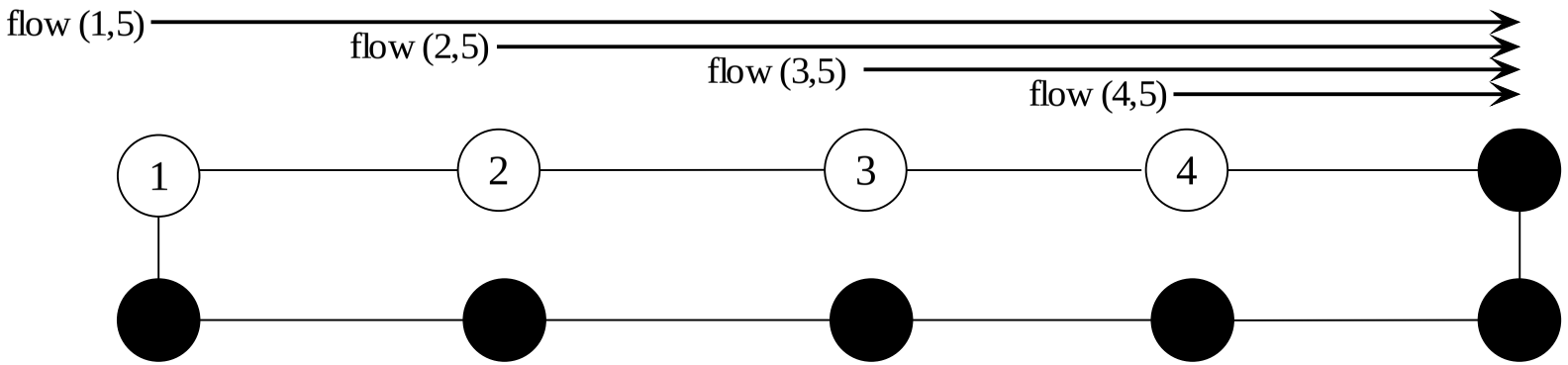
Scenario I: Balanced Parking Lot



- Parking Lot configuration
- Infinite Demand Flows (balanced or homogeneous inputs)

- Performance Issues:
  - RIAS fair rates (in steady state)
  - Convergence times
  - Others: number of messages, inter-message time, architecture complexity, …

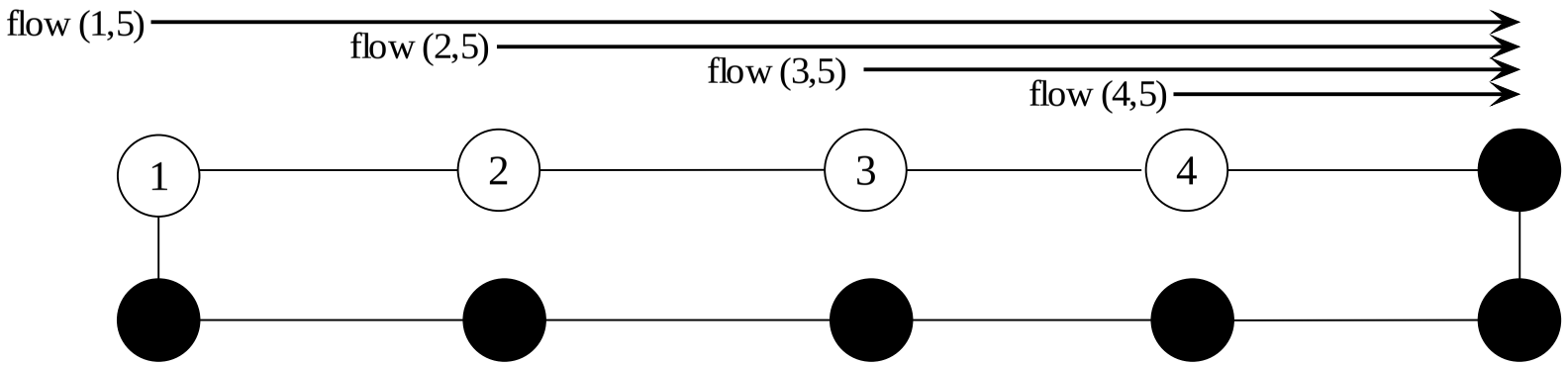
Scenario II: Unbalanced Parking Lot



- Parking Lot configuration
- Low- or variable-rate downstream flow (4,5). All others infinite demand

- Performance Issues:
  - Potential permanent oscillation
  - Throughput loss compared to RIAS fair rates

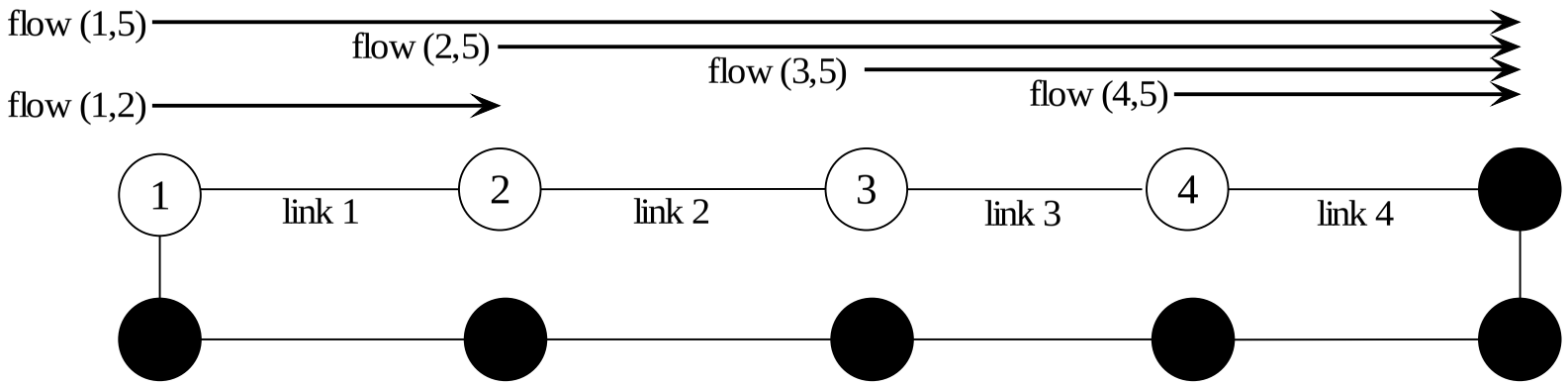
Scenario III: TCP+UDP Parking Lot



- Parking Lot configuration
- Flow (1,5) is aggregate TCP traffic. All others are infinite demand UDP traffic

- Performance Issues:
  - RPR should provide inter-node performance isolation (TCP traffic still obtains ¼ bandwidth)

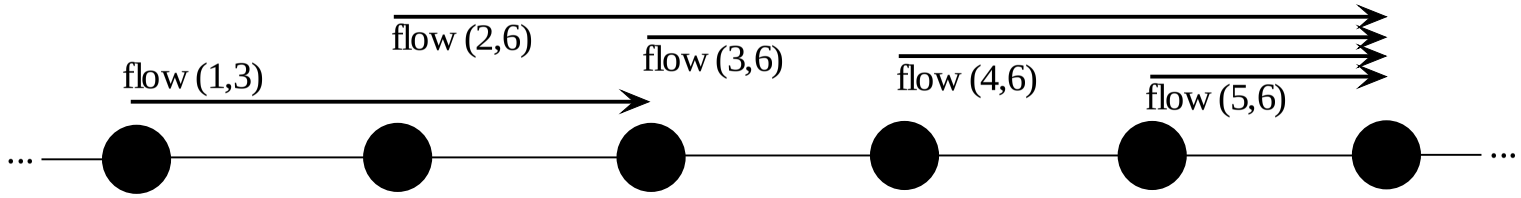
Scenario IV: Parallel Parking Lot



- "Parallel Parking Lot" configuration
- Balanced infinite demand traffic

- Performance Issues:
  - Fully exploit spatial reuse via per-destination queues and correct RIAS fair rate assignment

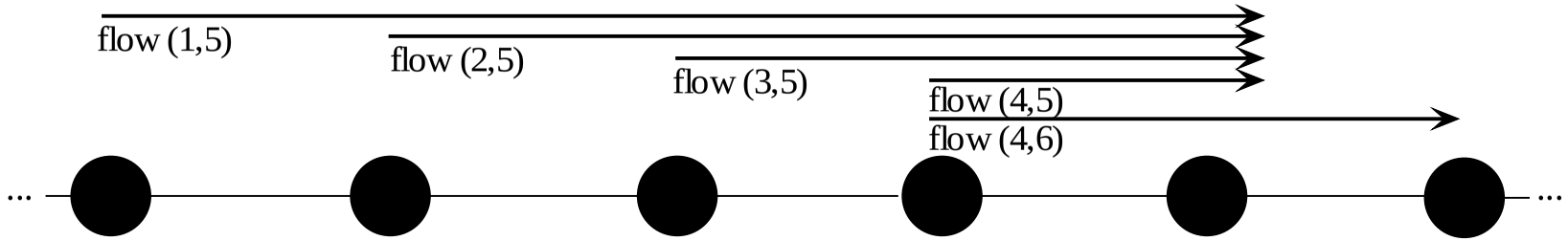
Scenario V: Upstream Parallel Parking Lot



- Parallel flow (1,3) originates upstream
- Balanced infinite demand traffic

- Performance Issues:
  - RIAS fair rates are achieved in two steps
    - Throttle flow (2,6) to ¼
    - Increase flow (1,3) rate to ¾
  - Ability to achieve these rates and convergence time
  - Results in "unbalanced traffic" despite homogeneous inputs

Scenario VI: Two Exit Parking Lot



- Parallel flow originates upstream
- Balanced infinite demand traffic

- Performance Issues:
  - RIAS fair rates are 1/8 for flows (4,5) and (4,6), ¼ for others
  - Not 1/5 for all

Enhanced Algorithms

- We developed DVSR: Distributed Virtual Time Scheduling in Packet Rings
- Key ideas
  - Use per-ingress byte counts from transit path traffic
  - Bound the local RIAS-fair rates
  - Advertise this bandwidth instead of my_rate and adapt (iterate)
- Tradeoff: improved convergence and better approximation of RIAS-fair at the cost of additional measurement and rate computation complexity
  - Have 1 Gb/sec network processor prototype
- I am NOT proposing DVSR as a standard!!

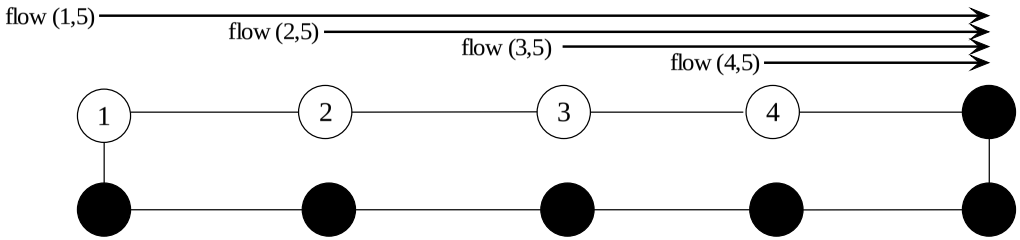
Simulation Experiments

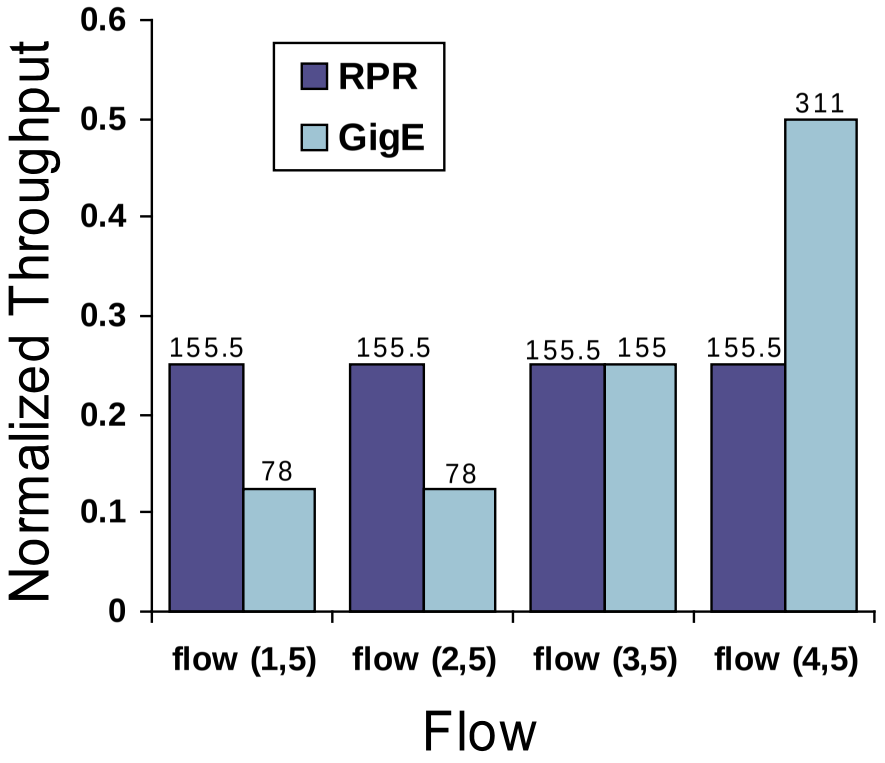
- Algorithms
  - OPNET modules – Gandalf, Aladdin, and GigE
    - All default settings
  - *ns-2* implementation of DVSR

- Scenario
  - 622 Mbps link capacity
  - 200 kByte buffer size
  - 1 kByte packet size
  - 0.1 msec link propagation delay
  - 1 msec feedback time

I. Fairness in the Parking Lot

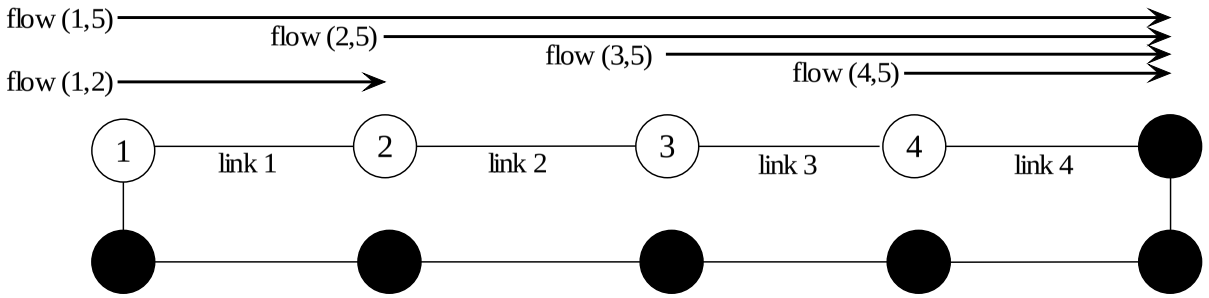
- Four constant-rate UDP flows sending at 622 Mbps
- RPR (Darwin, DVSR, Aladdin, …) provide RIAS fair shares
- GigE does not





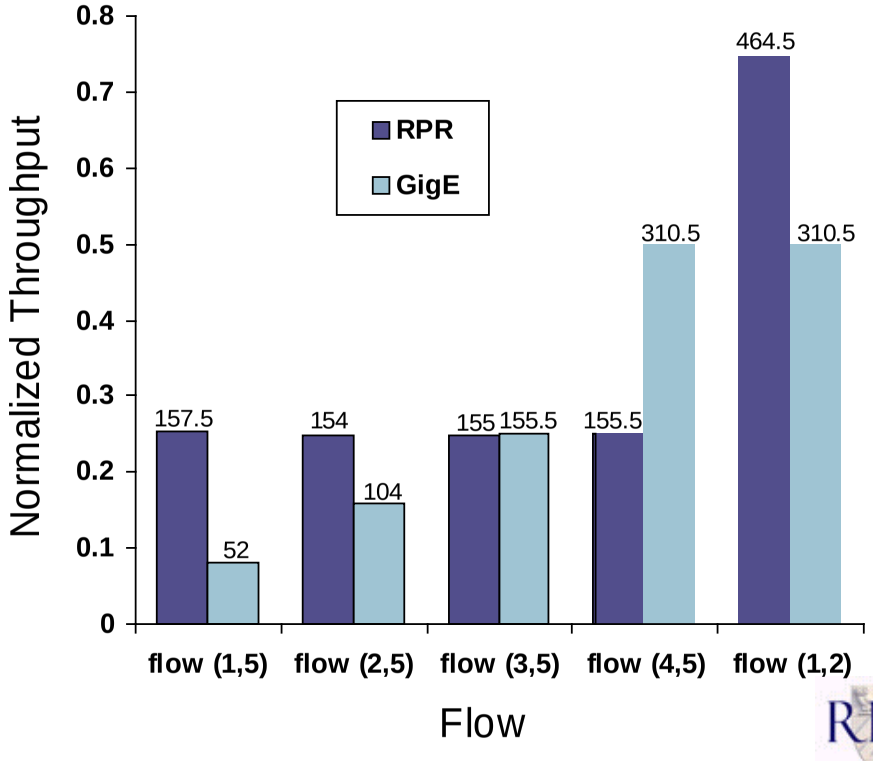
IV. Spatial Reuse in the Parallel Parking Lot

CBR UDP flows sending at the link capacity



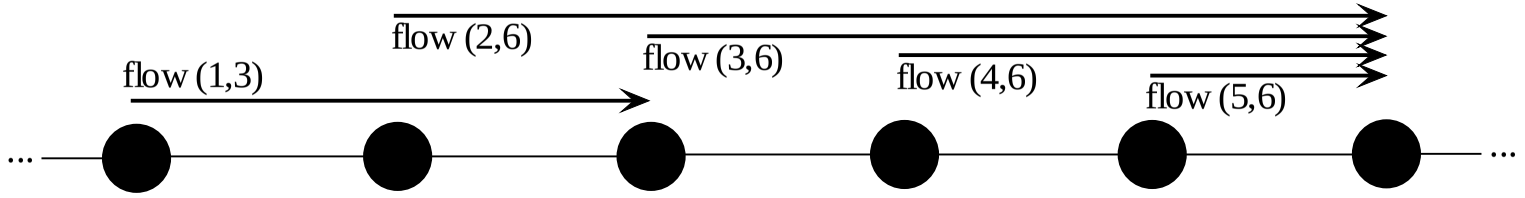
- RPR is within ±1% of RIAS fair rates
- GigE favors downstream flows & cannot achieve spatial reuse
- Darwin achieves only if using per-destination queues.

  Otherwise, flow (1,2) has 83% throughput loss, flow (1,5) has 50% throughput loss vs. RIAS fair rate

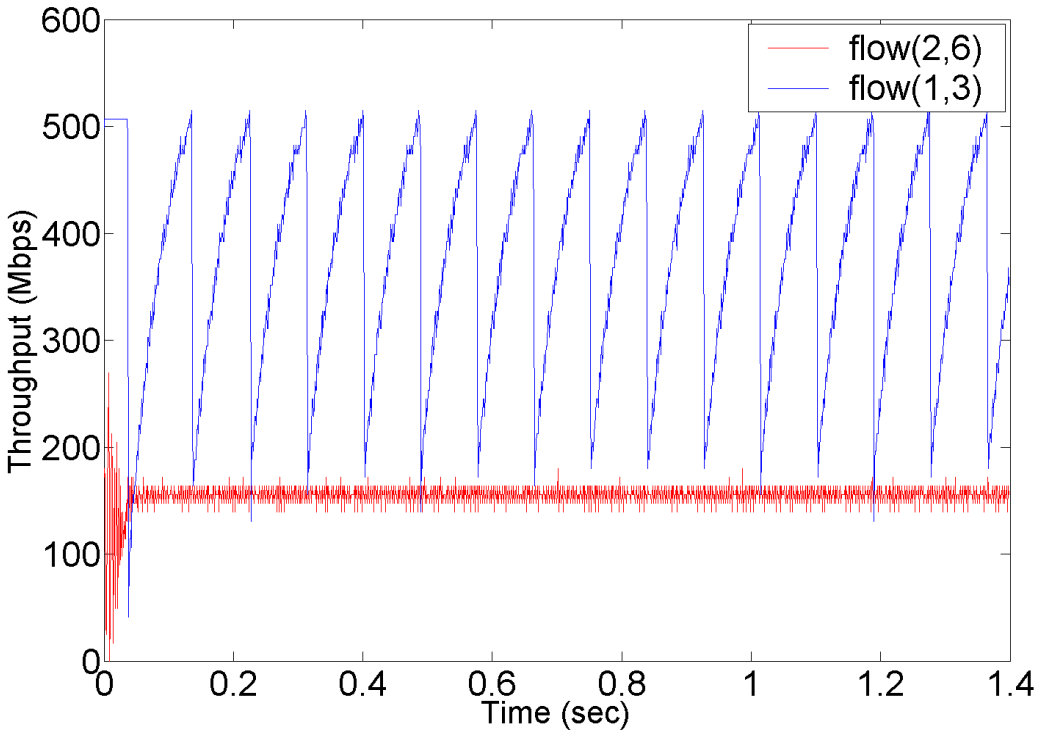


RICE

Scenario V: Unbalanced Traffic

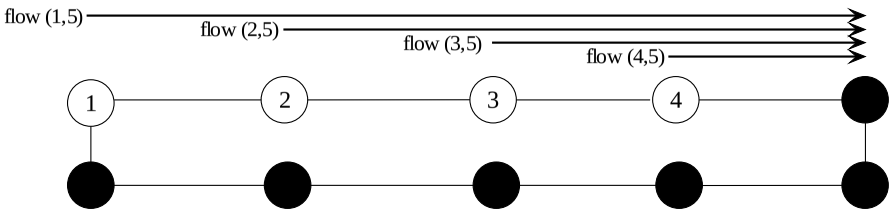


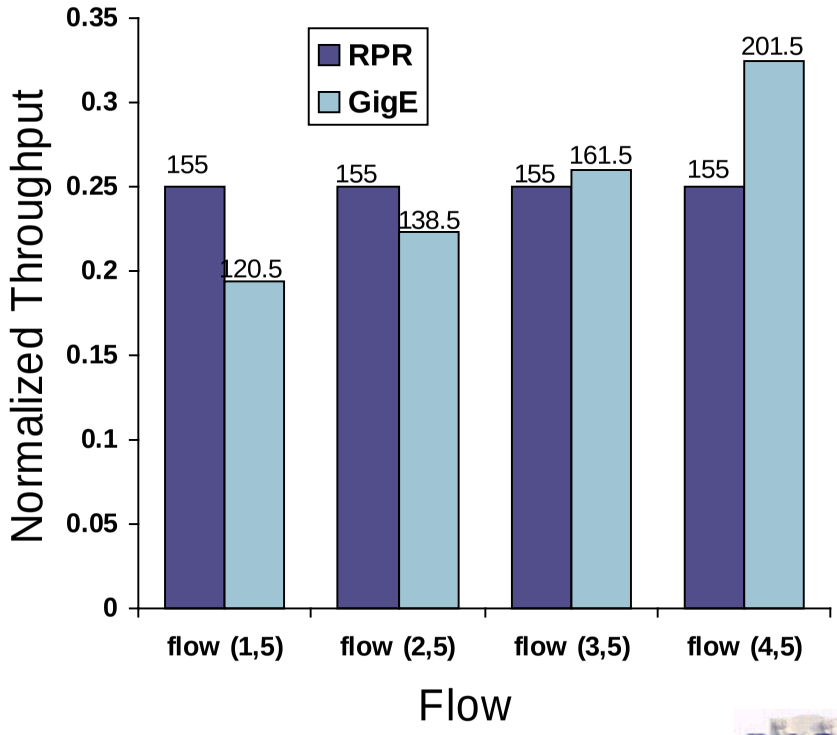
- Gandalf oscillation range is 0.25 to 0.75 and throughput loss is 14%
- DVSR loss is < 0.1%
- Many other scenarios can result in traffic imbalances and throughput losses



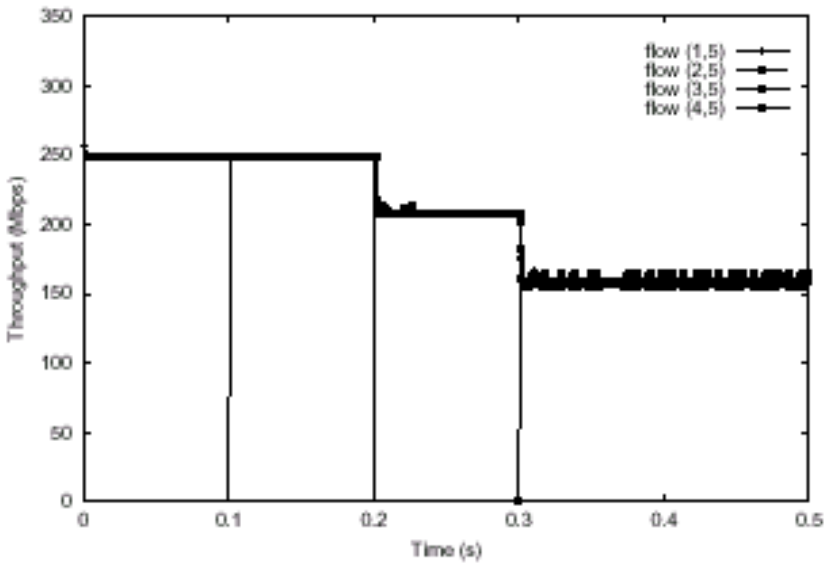
III. RIAS vs. Proportional Fairness for TCP Traffic

- Each flow =1 TCP micro flow (ftp/TCP Reno)
- Rate within ±1% of RIAS fair rates for 1 TCP micro-flow
- GigE tends to provide "proportional fair" rates

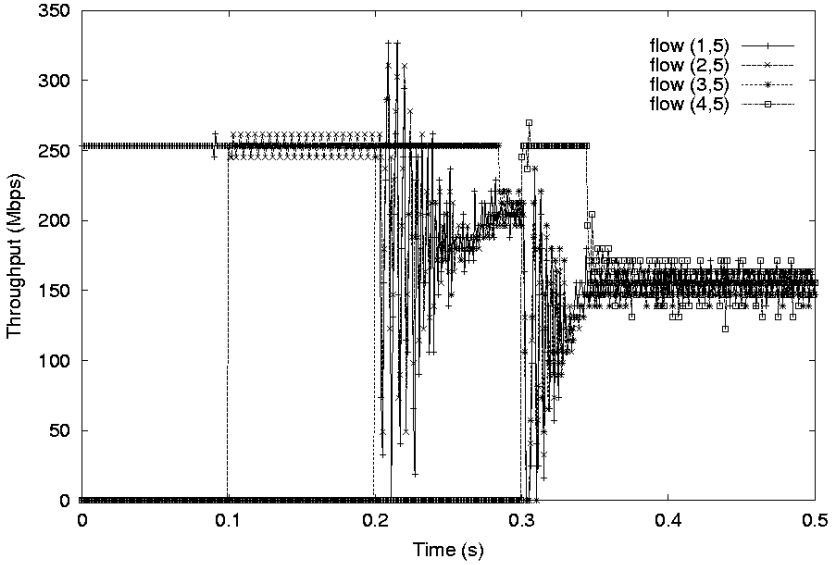




Scenario I: Convergence Time



DVSR



Gandalf

- CBR UDP flows with rate 0.4 (248.8Mbps)
- Flow(1,5), (2,5), (3,5), (4,5) begin transmission at times 0.0, 0.1, 0.2, and 0.3 seconds respectively
- Convergence time 0.2 msec for DVSR, 50 msec for Gandalf
- Richer feedback signal allows faster convergence

Conclusions (1/2)

- RIAS fairness as proposed reference model for RPR
- Benchmark scenarios for algorithm development
- Gandalf can permanently oscillate in a wide range even under balanced infinite demand (Scenario V)
  - Throughput loss is approximately 15% and is dependent on delays and filter settings
  - Root cause is unbalanced fair rates (vs. input rates)
  - Impact to TCP unknown
- Alladin has similar vulnerability
- Convergence times of DVSR significantly faster than Gandalf
  - Throughput and response time improvements

Conclusions (2/2)

- Recommendation: leave feedback specification liberal

  Ex. "Nodes should feedback an approximation of the RIAS fair rates given their collected information"

  1. Then Darwin as well as DVSR-like enhancements are both standard compliant.
  2. Vendors can operate in wide spectrum of performance (convergence and throughput) and complexity (what is measured and computations on measurements)
  3. GigE cannot derail RPR by claiming "Darwin oscillates permanently within the entire link bandwidth"

- Non-recommendation: standardize DVSR

How We Hope To Contribute

- Performance analysis and RPR comparison with GigE and RPR enhancements
- Open source ns implementation of Darwin
- Open source ns implementation of DVSR
- Help refining drafts

Backup Slides

- Precise definition of RIAS fair

Step I: Local Fairness

- Label nodes 1, …, N and links 1, …, N-1
- r_{ij} is the traffic demand between nodes i and j at a particular time instant
- r_i^n is the Ingress Aggregated traffic from ingress node i at link n

$$r_i^n = \sum_{j>n} r_{ij}$$

- The locally fair allocation on link n is

$$R_i^n = \max\_\min_i (C, r_1^n, r_2^n, ..., r_n^n)$$

Footnote on max_min

- What is $\text{max\_min}_i(\ )$?
  - The "textbook" definition of (locally) fair
  - Would be achieved by fair queueing if fair queueing was performed on ingress aggregates
  - Can write down the exact computation [BerGal92,p527]

Step II: Ingress Fairly Sub-allocates Per-link Bandwidths

- $R_{ij}^{n} = \max\_\min_{j}(R_{i}^{n}, r_{i,n+1}, r_{i,n+2}, \ldots, r_{i,j}, \ldots, r_{i,N})$
- Ingress has bandwidth R_{i}^{n} on link *n* and divides it fairly among flows traversing *n*
- End-to-End rate is the bottleneck rate

$$r_{i,j} = \min_{n} R_{ij}^{n}$$

Step III: Iterate

- There may be further bandwidth available for spatial reuse
  - Due to multiple congestion points
- Iterate process such that all excess capacity is fairly reclaimed
- Set new capacity to all unallocated capacity

$$C^n = C^n - \sum_{ij} R_{ij}^n$$

- Go to Step I

Backup Slides

- Details of DVSR

DVSR Challenge

How to know the local IA-fair rates without actually doing fair queueing and measuring them?

(which would make for a complex transit path)

Solution

- Compute a proxy of virtual time using per-ingress byte counts
  - Computing exact virtual time is well known to be complex
  - We derived a simple bound using arrival counts
- Communicate virtual-time rate upstream
- Ingress nodes compute minimum rate on the path and converge to RIA fair rates

DVSR Protocol
(as implemented in the simulator and testbed)

- Scheduling
  - SP or Darwin
- Computation of feedback signal
  - Byte count for each ingress node - Lower bound of virtual time
  - Observation: Minimal increase in *v(t)* if all packets arrive at the beginning of the interval
  - $l_1 <= l_2 <= \ldots <= l_k$

```
if (b < 1) {F = l_k/CT + (1 − b)}
else {
   i=1
   F = 1/k
   Count = k
   Rcapacity = 1
   while ((l_i/CT < F )&&(l_k/CT ≥ F)) {
      Count- -
      Rcapacity -= l_i/CT
      F = Rcapacity / Count
      l_i = l_i+1
   }
}
```

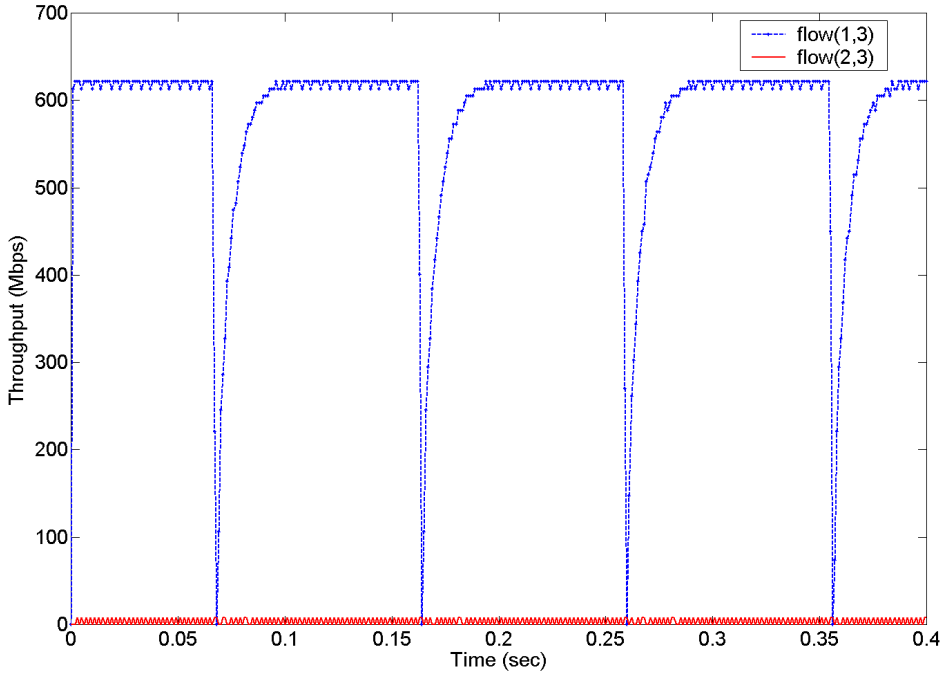
DVSR Properties

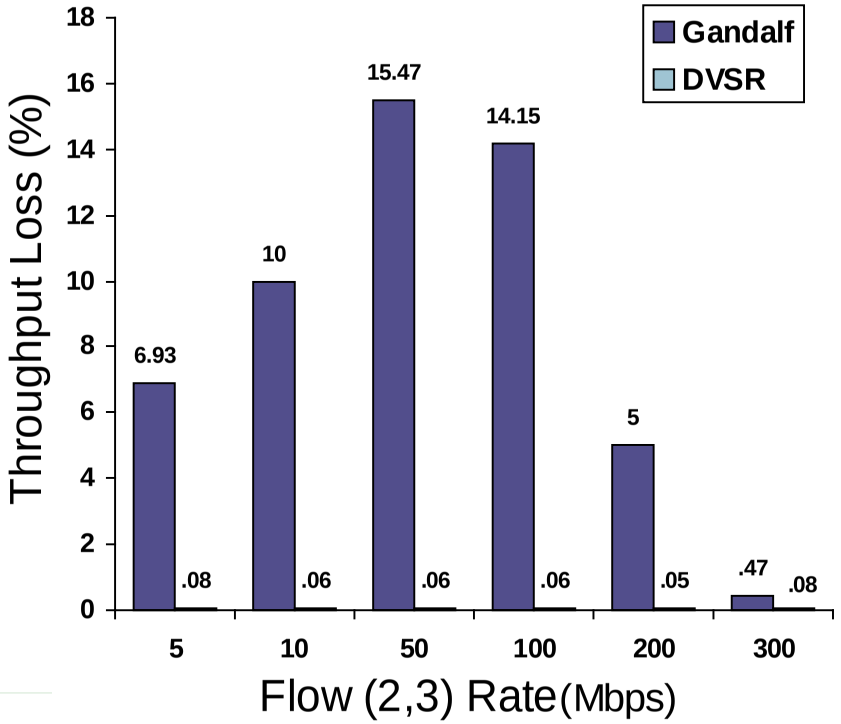
- Complexity
  - Ordering rates is O(klogk) (k is at most N/2). Performed every control update time (.1 msec minimum)
    - Developing approximations to avoid
  - Rate limit computation
    - Sub-allocation of per-link IA fair rates to source-destination pairs
- Feedback signal
  - Darwin-compatible replacement to my_rate
- Fairness
  - Derived a worst-case bound on unfairness

Backup Slides

- Details of Unbalanced Traffic Simulations

Gandalf with Unbalanced Traffic





- Gandalf: flow (1,3) permanent oscillation in full range of link capacity due to low *my_rate*
- Consequence: throughput degradations up to 15% in this case

RICE

Alladin with Unbalanced Traffic

- Parking lot scenario
- A single flow demanding 700 Mbps (CBR)
- A number of small downstream flows demanding 15 Mbps (CBR)
  - Packet size 100 bytes