Institut für
Kommunikationsnetze

Fairness Benchmarking of MACs

Harmen R. van As, Günter Remsak, Jon Schuringa

Vienna University of Technology, Austria

vas_benmac_03

Goal and Content

Goal: Finding an algorithm to determine the maximal individual node throughputs and at the same time fulfilling bottleneck-link fairness

Content:

- Two definitions for bottleneck fairness
- Corresponding fairness algorithms and examples
- Two traffic scenarios

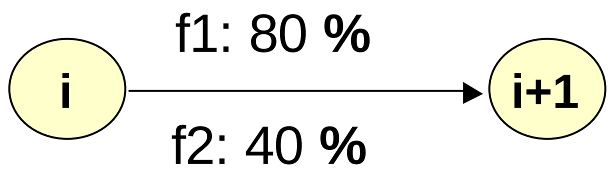
Local Fairness Definitions

1. Flow rates on bottleneck are proportionally reduced by the total amount of offered traffic for that bottleneck link

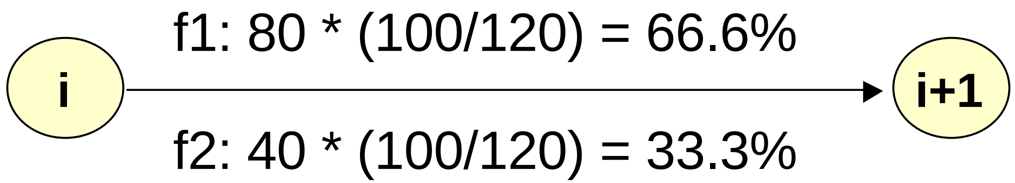
2. Flow rates on bottleneck are proportionally reduced by the total number of connections on bottleneck link

Example

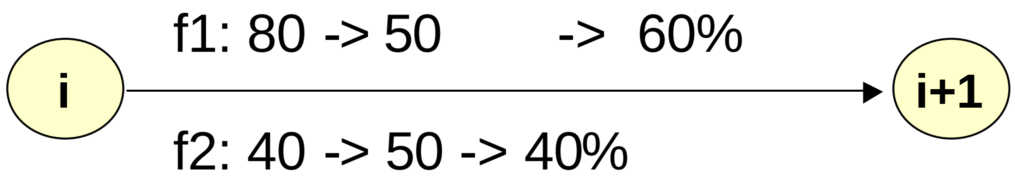
- 2 flows, 1 link:



- Definition 1:



- Definition 2:



Definitions

Given:

- Number of nodes — N
- Requested rate from node i to node j — $r_{i,j}$

Calculated:

- Flow on link i — f_i
  Sum of all requested rates passing link i
- Number of demand flows passing link i — nd_i
- Remaining capacity on link i — rc_i
  Link capacity minus the sum of all allowed rates passing link i
- Allowed rate from node i to node j — $ar_{i,j}$
  Rate calculated by the algorithms

Algorithm for Fairness Definition 1

Set: $rc_i=1;$ // init remaining capacity

Step 1: for all links: calculate flow on link i: f_i

Step 2: if ($rc_i < f_i$) // condition for a bottleneck

take always the highest overloaded bottleneck: $\min(rc_i/f_i)$

bottleneck link: indicated by index b

else $ar_{i,j}= ar_{i,j}+ r_{i,j}$; stop;

Step 3: for all flows passing this bottleneck set: $ar_{i,j}=rc_b/f_b \times r_{i,j}$ and $r_{i,j}=0$

Step 4: calculate remaining capacities rc_i of all links; goto **Step 1**;

vas_benmac_03

Algorithm for Fairness Definition 2

Set: $rc_i=1$;

Step 1: for all links: calculate flow on link i: f_i

Step 2: if ($rc_i < f_i$) // condition for a bottleneck

take always the highest overloaded bottleneck: $\min(rc_i/nd_i)$

bottleneck link: indicated by index b

else $ar_{i,j} = ar_{i,j} + r_{i,j}$; stop;

Step 3: for all flows passing this bottleneck:

if ($rc_b/nd_b > r_{i,j}$)

$ar_{i,j} = r_{i,j}$; $nd_b = nd_b - 1$; $r_{i,j}=0$;

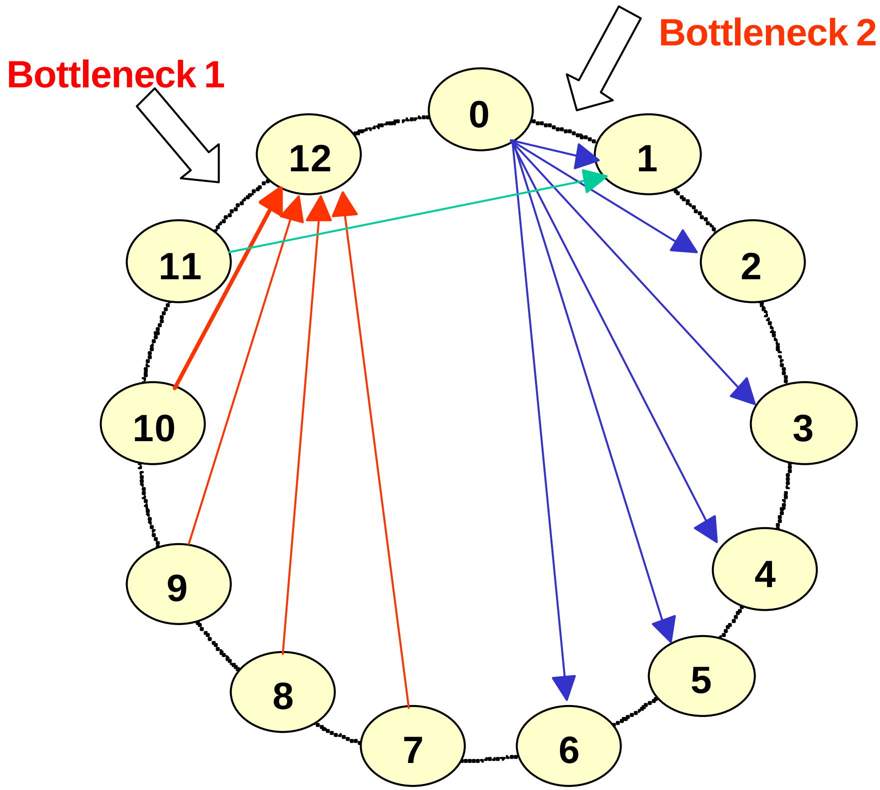
calculate remaining capacities rc_i of all links;

goto **Step 1;**

else $ar_{i,j} = rc_b/nd_b$; $r_{i,j}=0$;

Step 4: calculate remaining capacities rc_i of all links; goto **Step 1**;

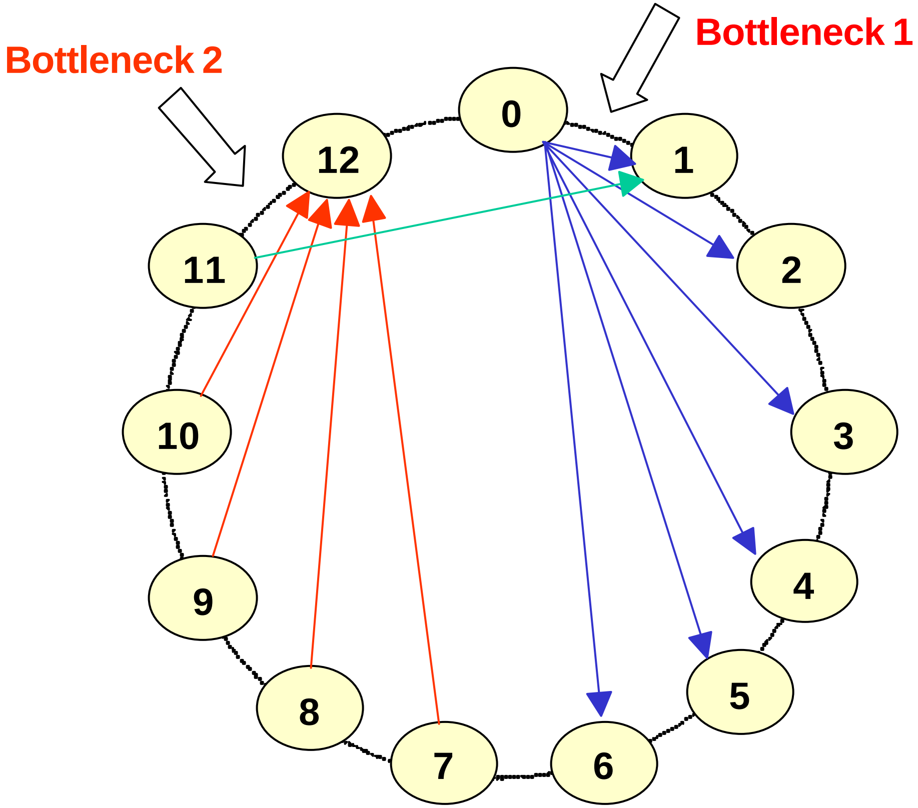
Example: Fairness Definition 1



Source	Sink	Rate	Fair
0	1	0.01	0.01
0	2	0.01	0.01
0	3	0.01	0.01
0	4	0.01	0.01
0	5	0.01	0.01
0	6	0.01	0.01
7	12	0.1	0.071
8	12	0.1	0.071
9	12	0.1	0.071
10	12	0.1	0.071
11	1	1	0.71

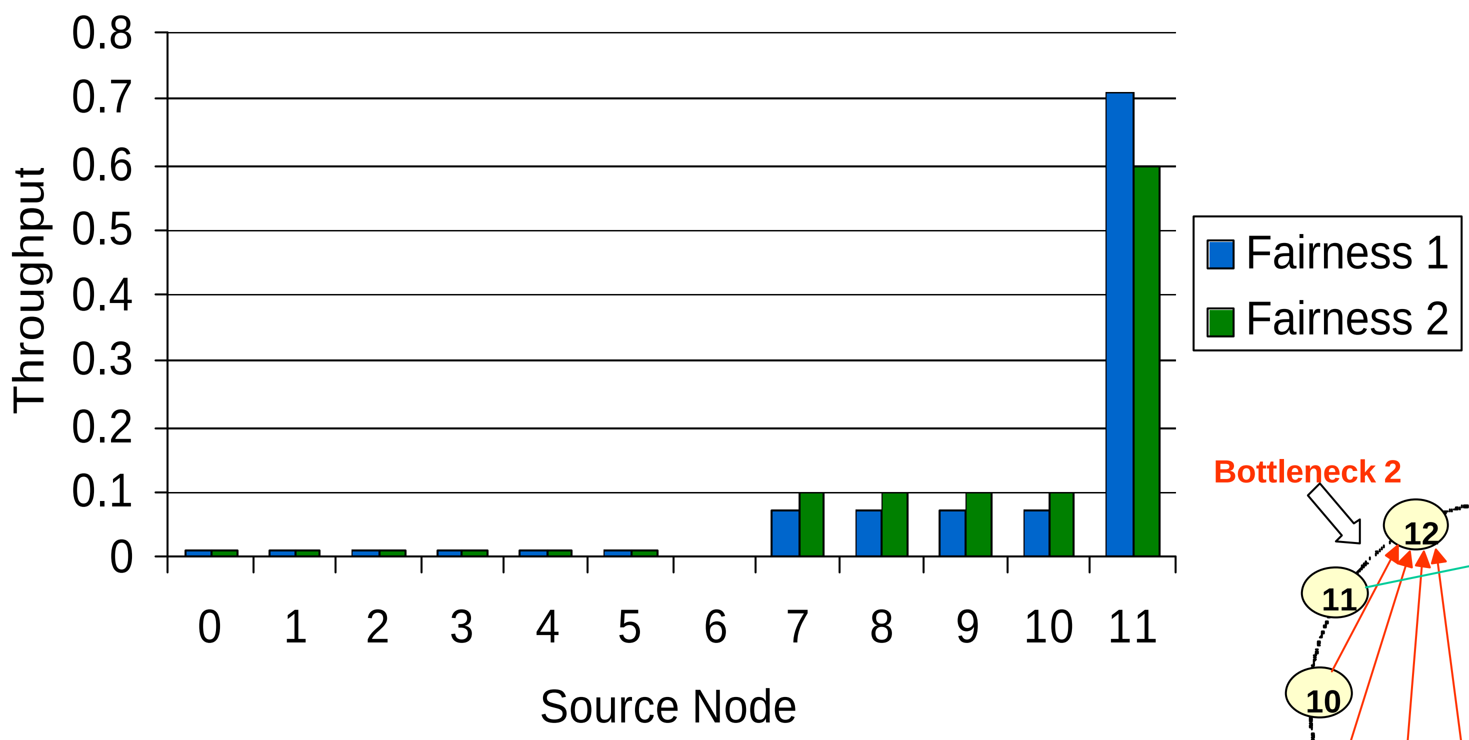
vas_benmac_03

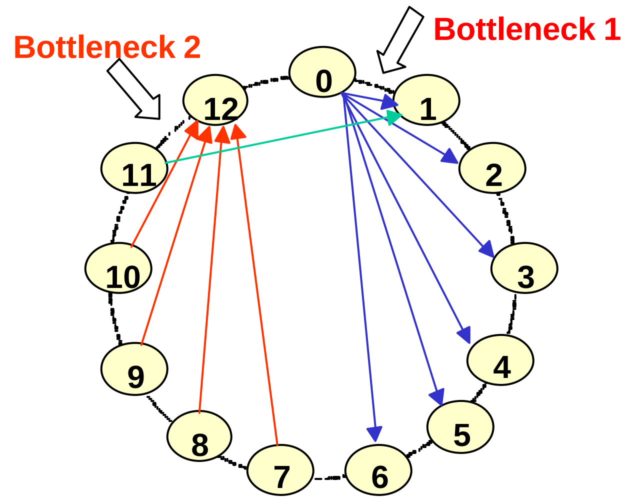
Example: Fairness Definition 2



Source	Sink	Rate	Fair
0	1	0.01	0.01
0	2	0.01	0.01
0	3	0.01	0.01
0	4	0.01	0.01
0	5	0.01	0.01
0	6	0.01	0.01
7	12	0.1	0.1
8	12	0.1	0.1
9	12	0.1	0.1
10	12	0.1	0.1
11	1	1	0.6

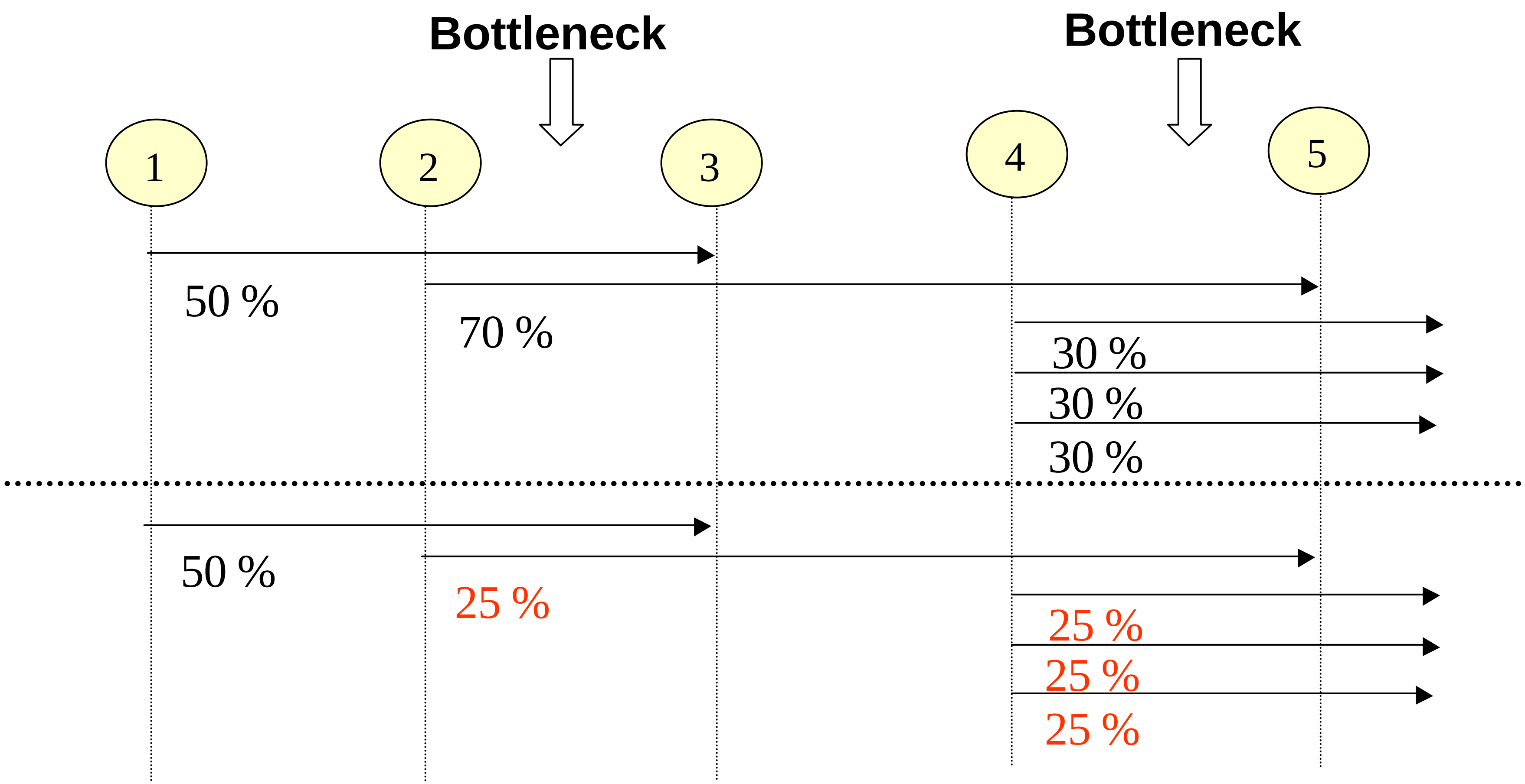
Throughput





vas_benmac_03

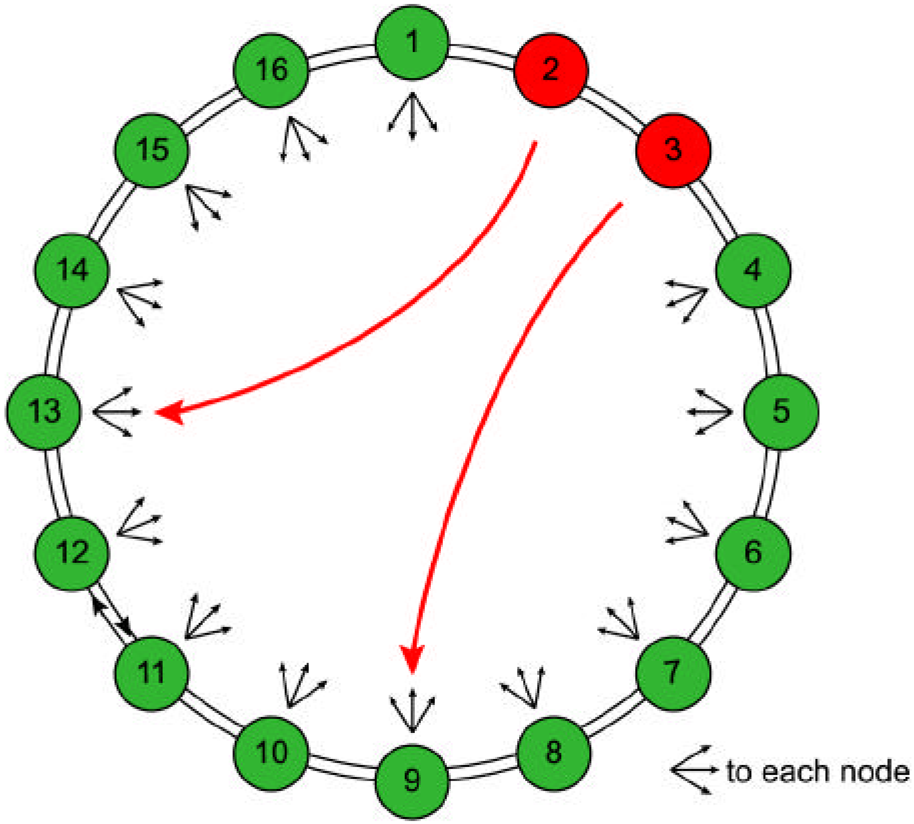
Example: Fairness Definition 2



Some bottlenecks can be resolved by resolving other bottlenecks

vas_benmac_03

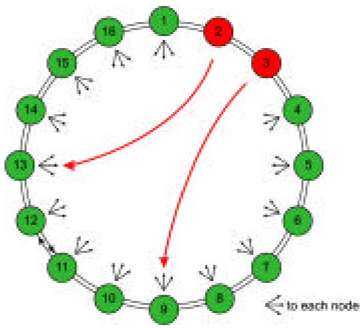
Scenario 1

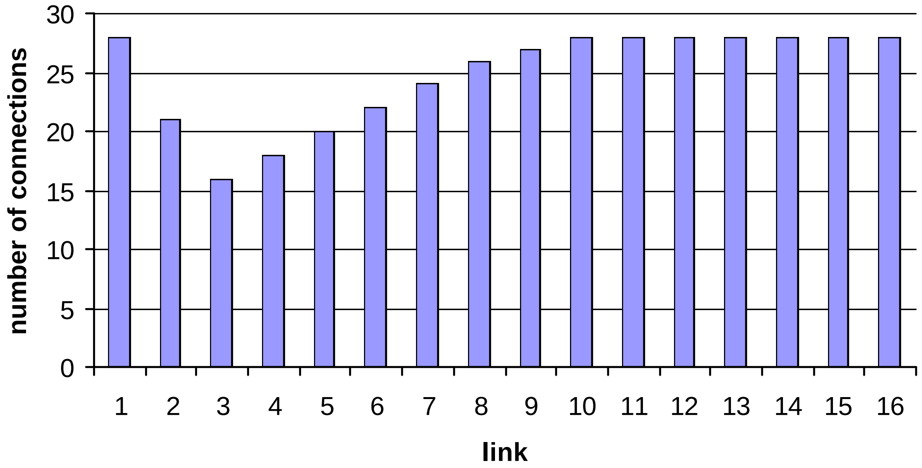


Uniform traffic
Saturated sources
16 nodes

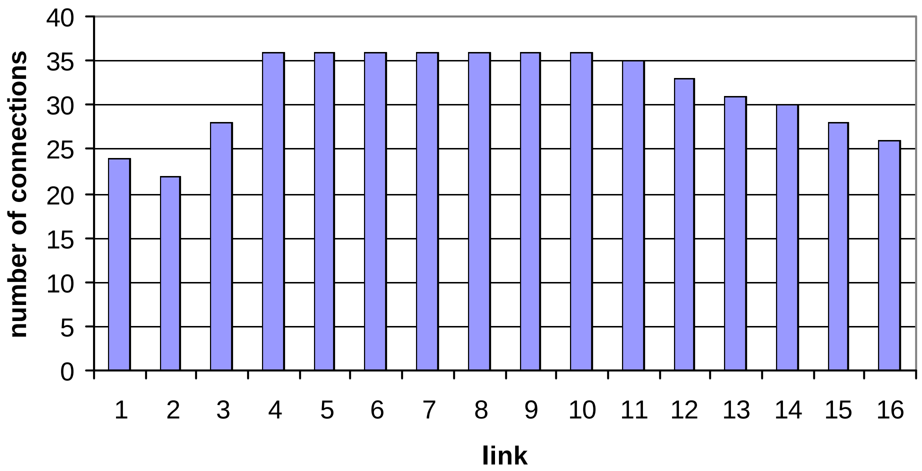
Number of Connections per Bottleneck Link

Scenario 1





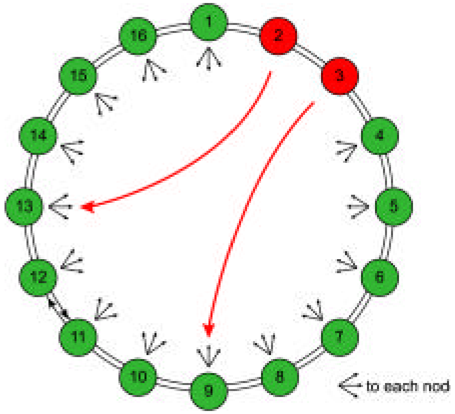
Ring 0

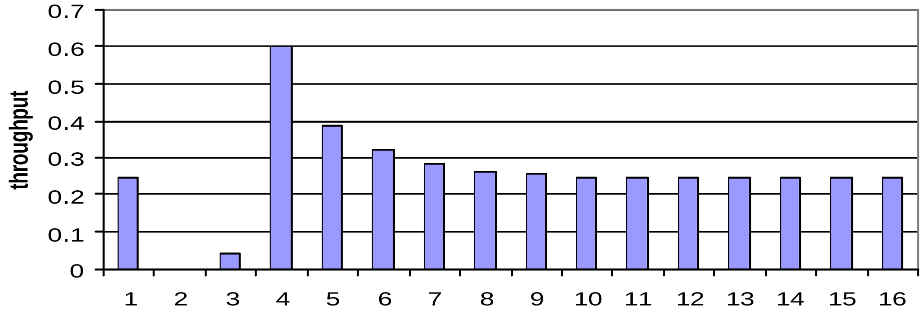


Ring 1

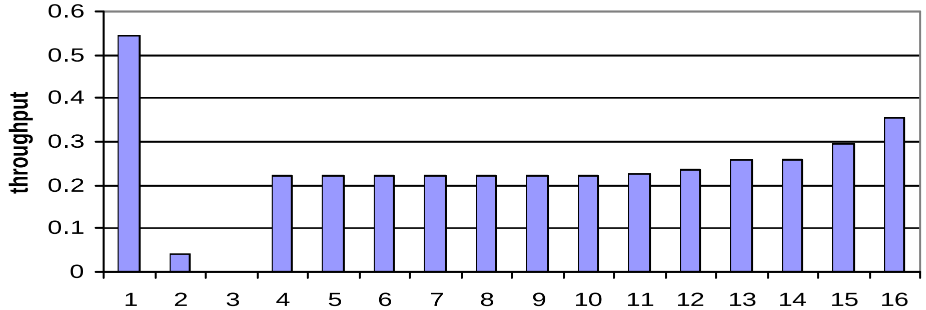
Throughput per Node

Scenario 1

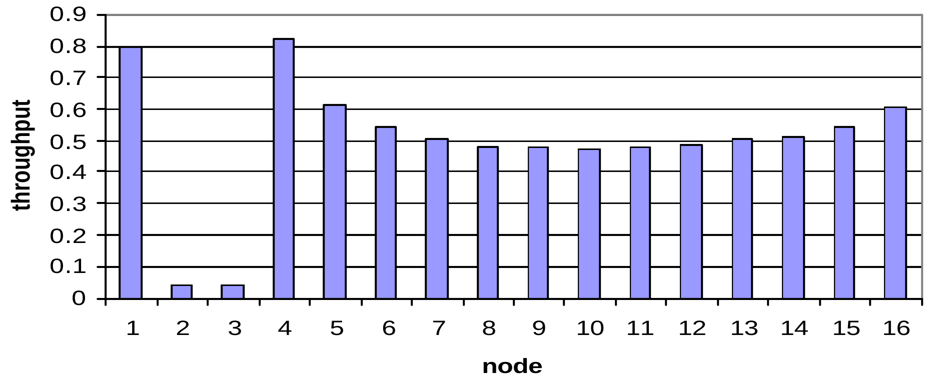




Ring 0



Ring 1

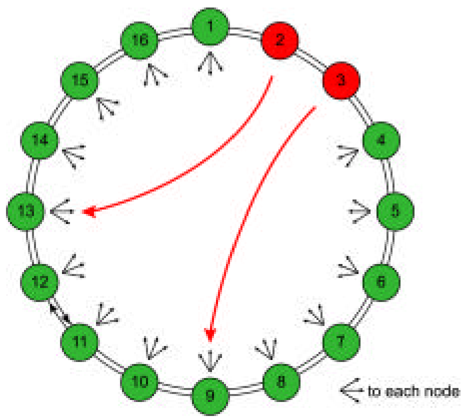


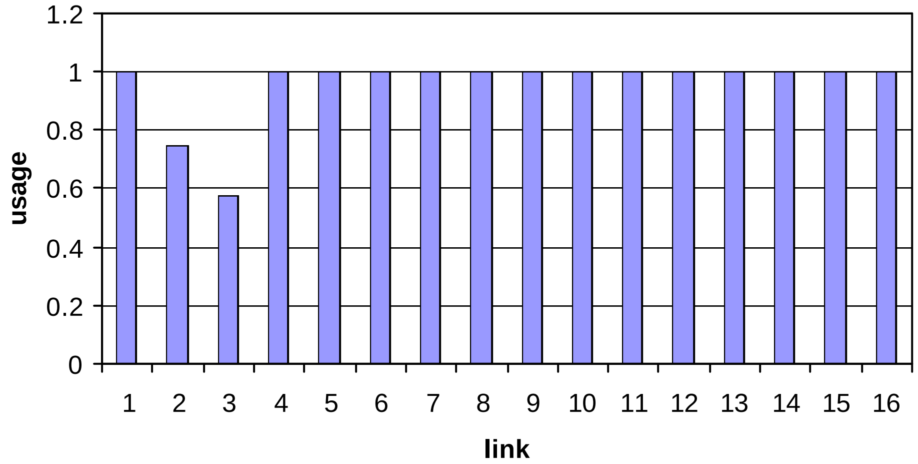
Ring 0 + Ring 1

vas_benmac_03

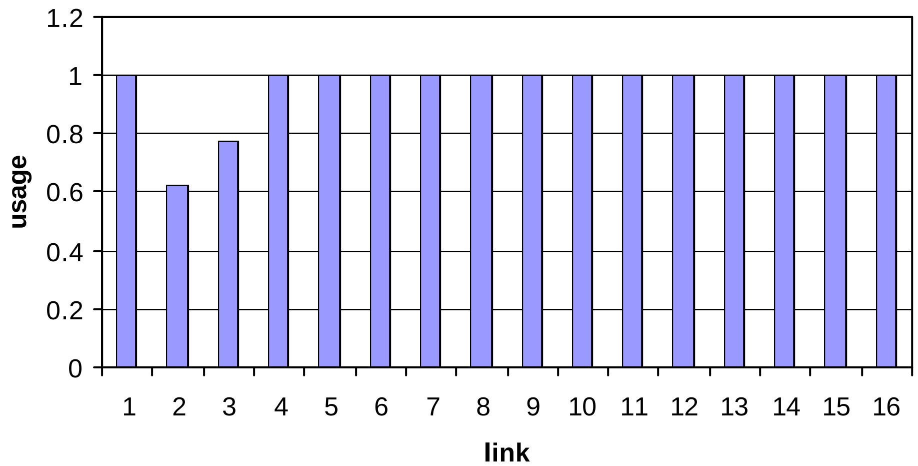
Link Usage

Scenario 1





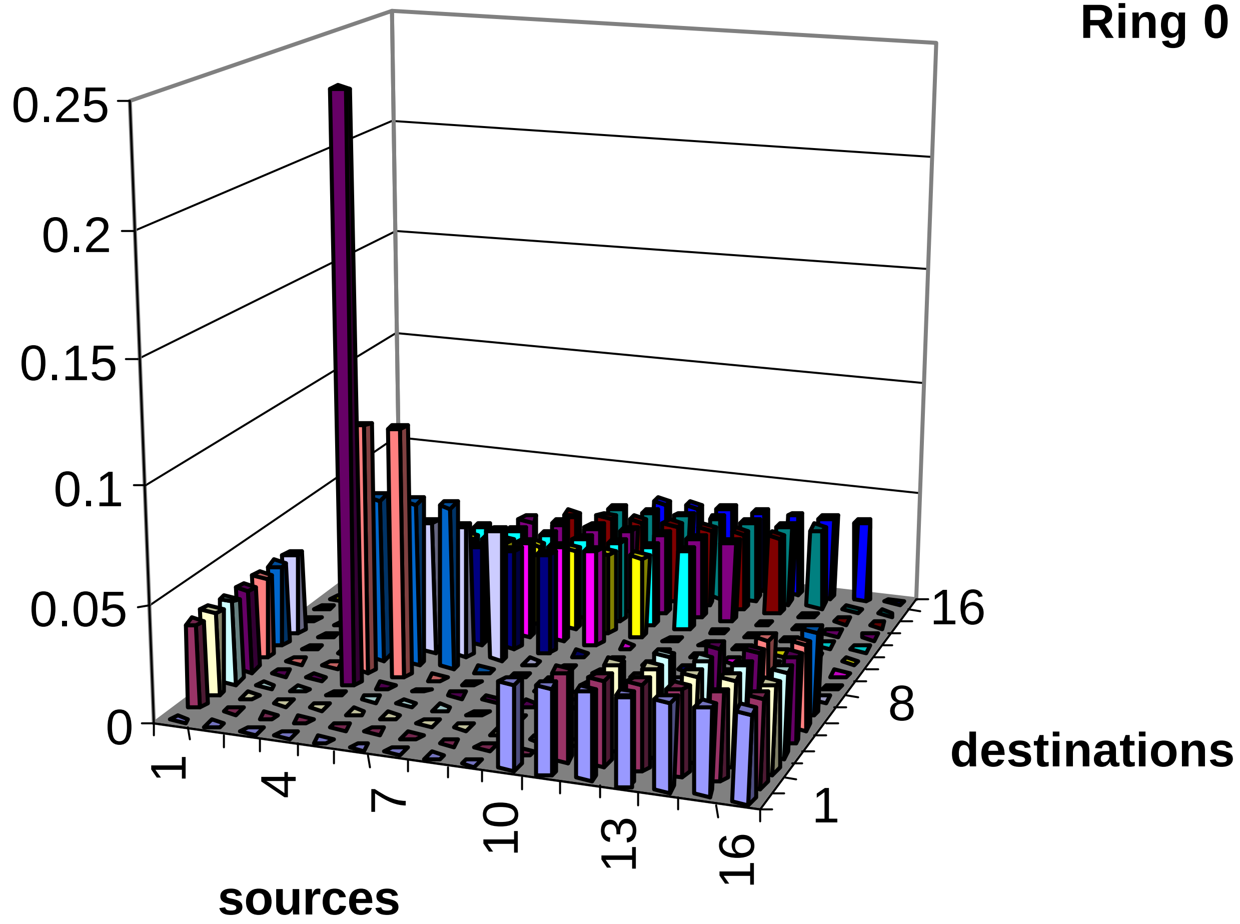
Ring 0



Ring 1

Throughput per Source/Destination Pair

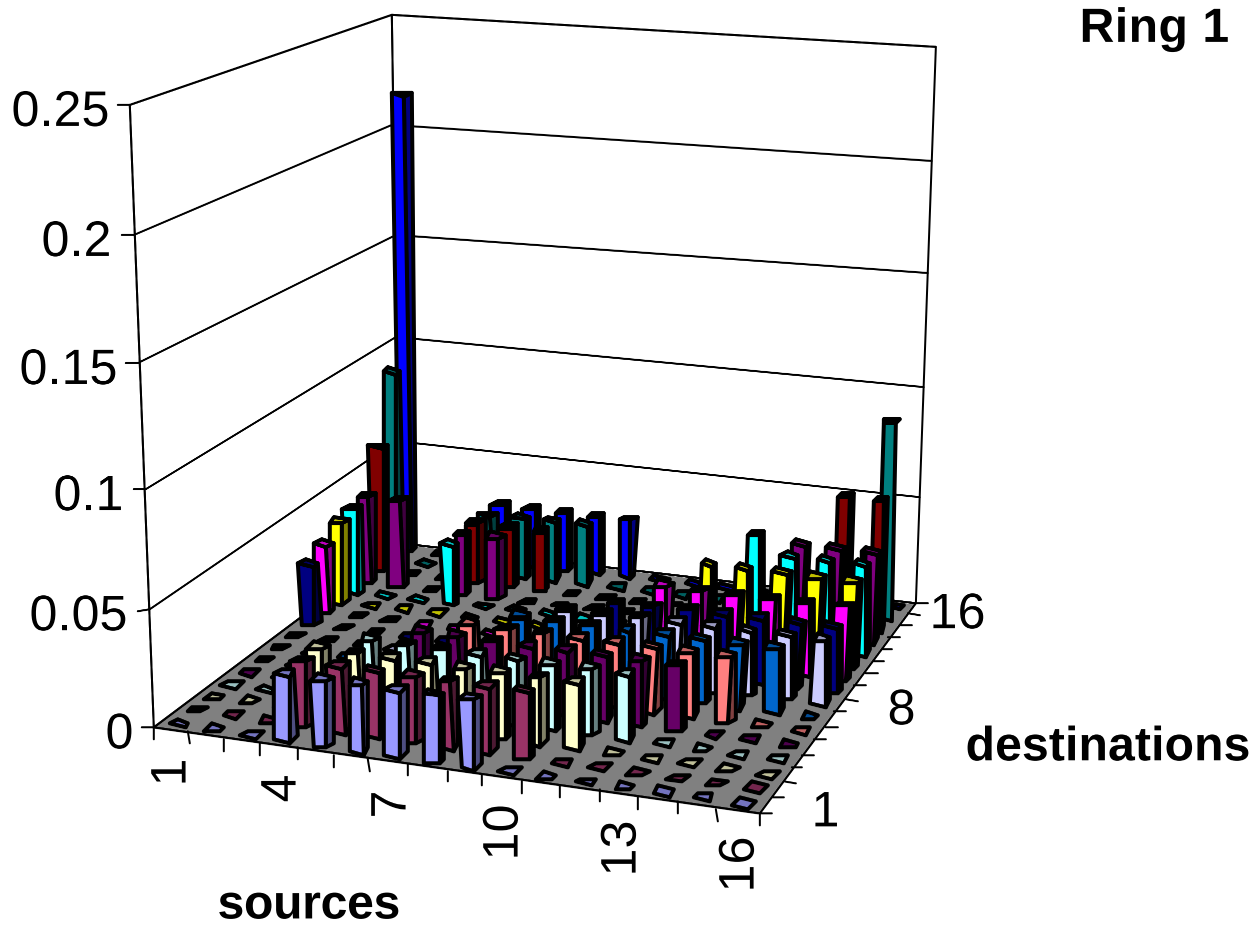
Scenario 1



vas_benmac_03

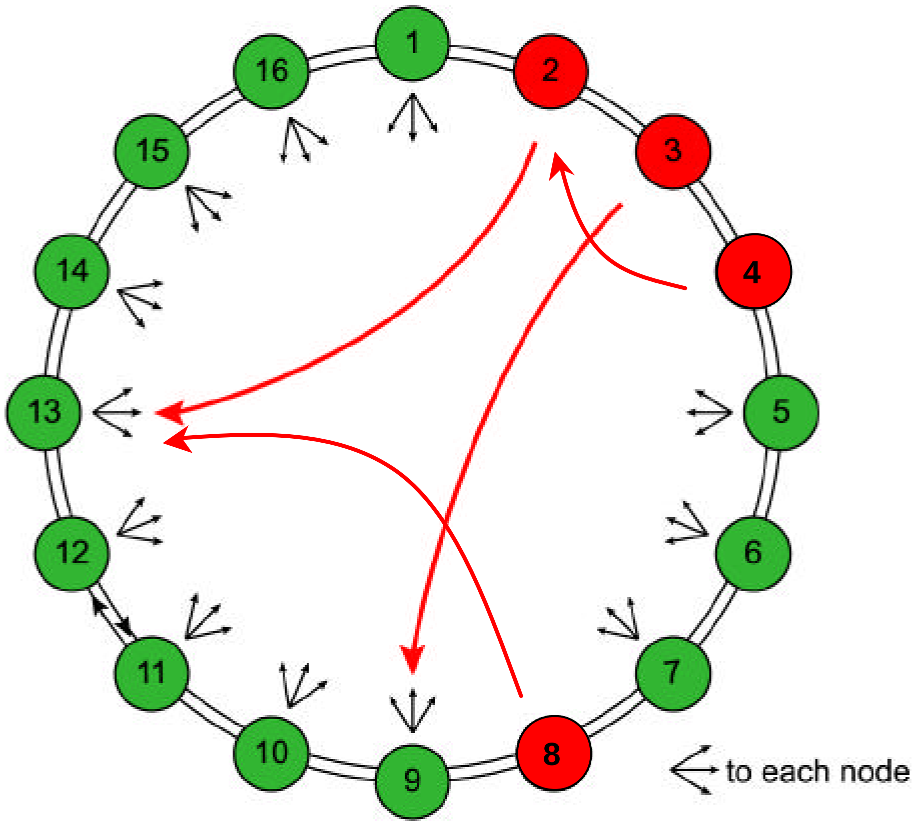
Throughput per Source/Destination Pair

Scenario 1



vas_benmac_03

Scenario 2

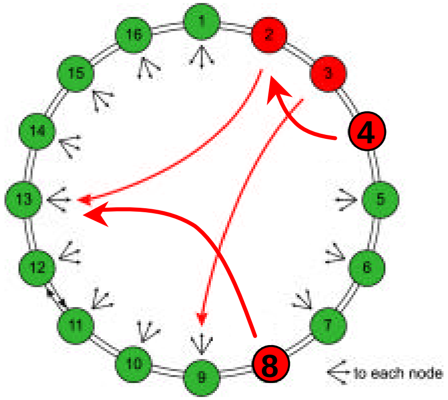


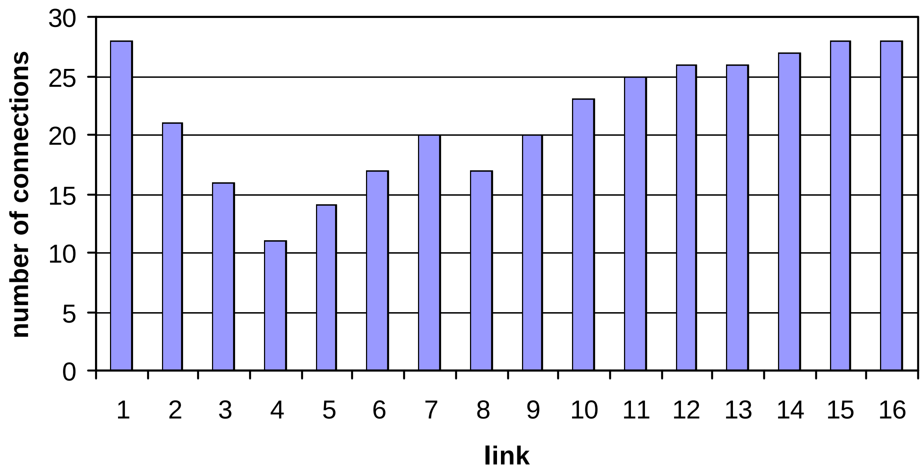
Uniform traffic
Saturated sources
16 nodes

vas_benmac_03

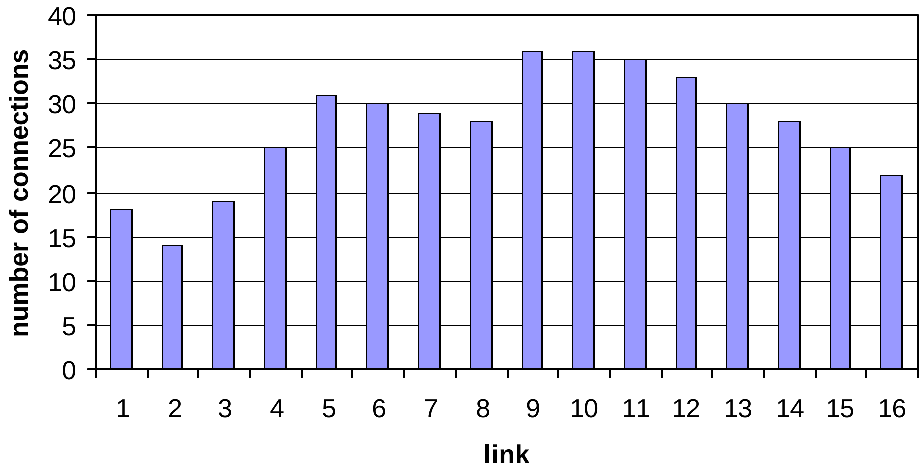
Number of Connections per Bottleneck Link

Scenario 2





Ring 0

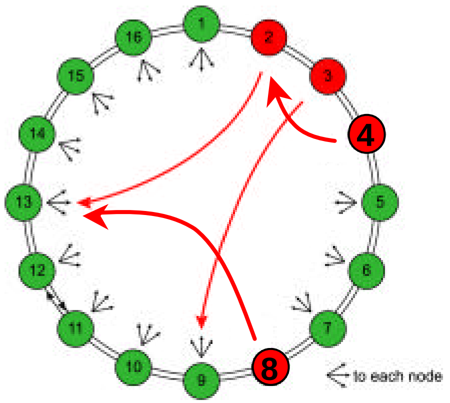


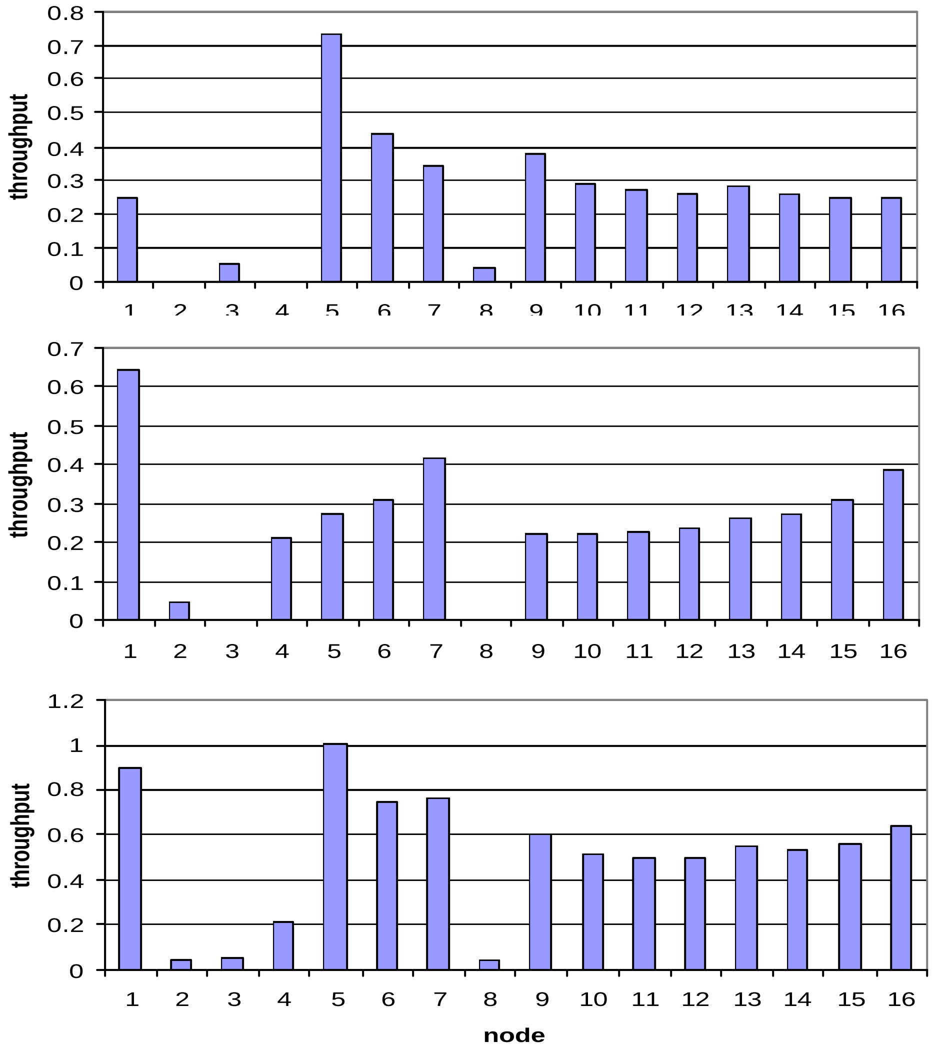
Ring 1

vas_benmac_03

Throughput per Node

Scenario 2





Ring 0

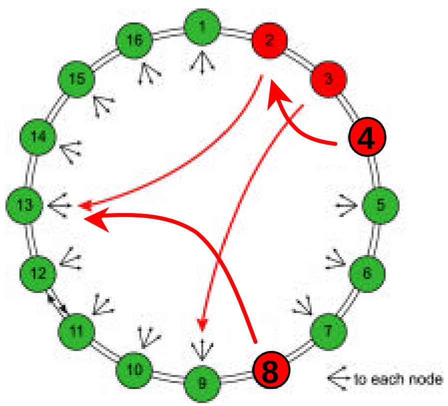
Ring 1

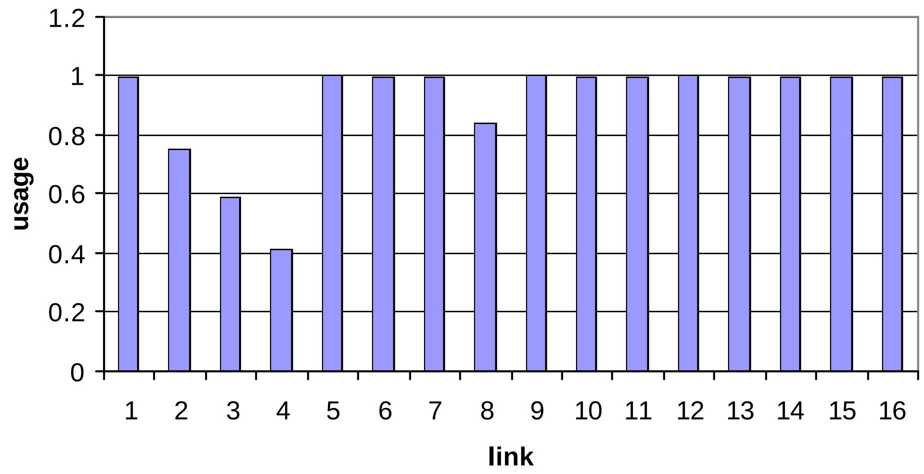
Ring 0 + Ring 1

vas_benmac_03

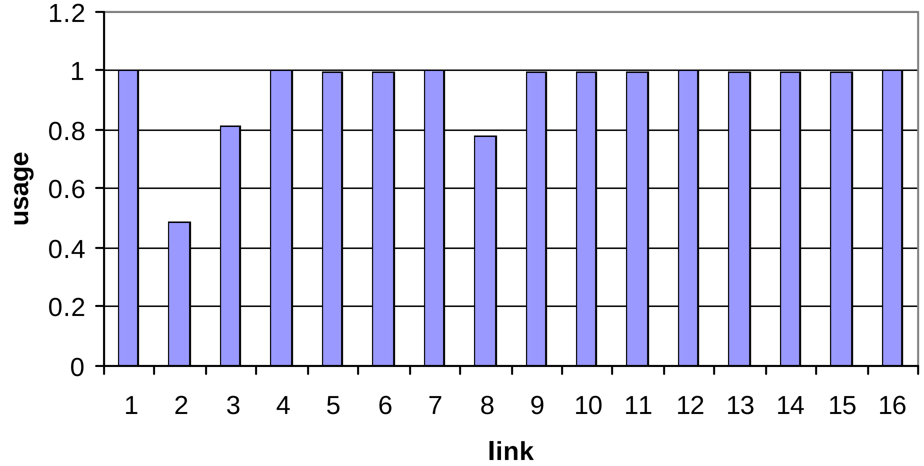
Link Usage

Scenario 2





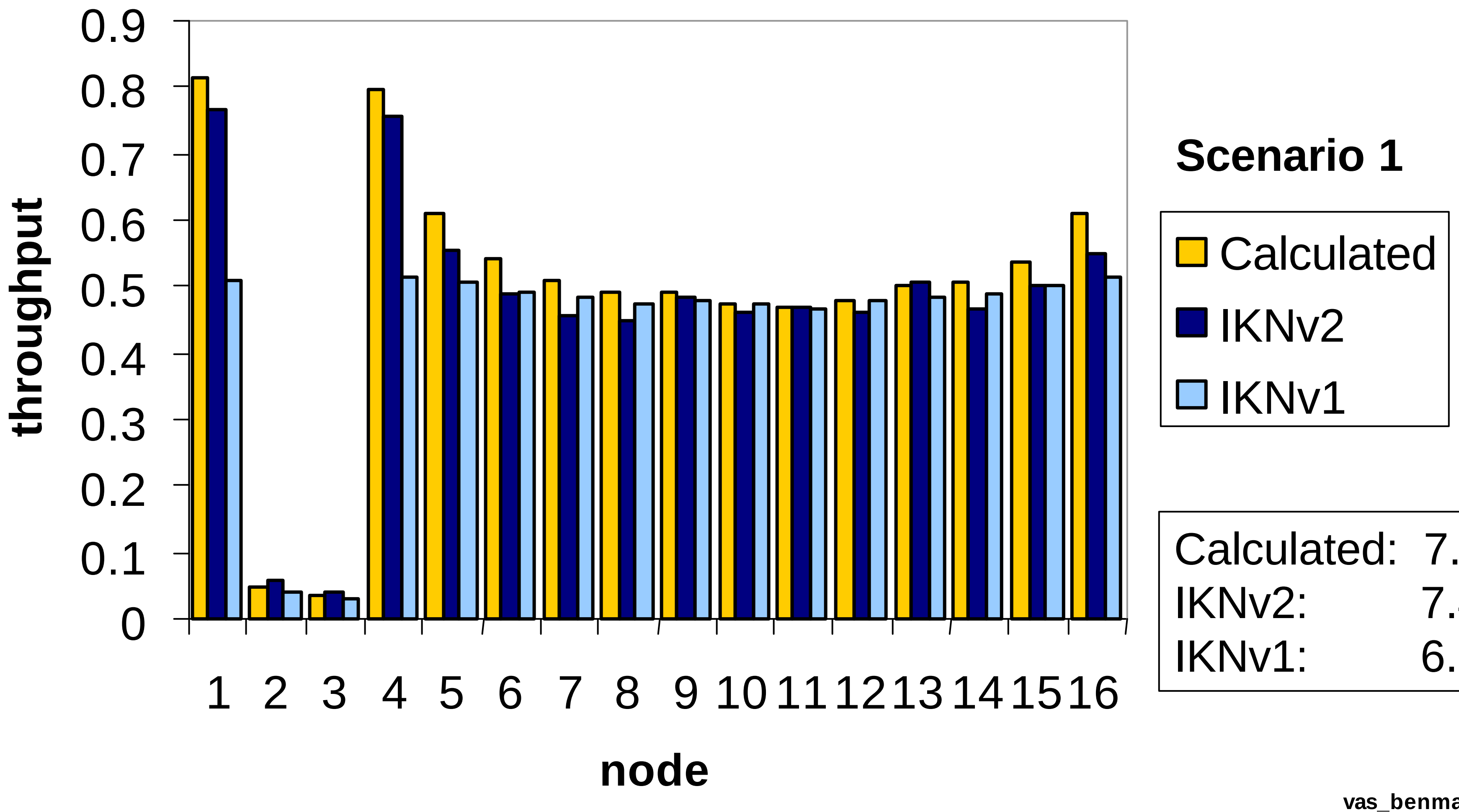
Ring 0



Ring 1

Throughput Comparison

Cyclic Reservation MAC : IKNv1 (July 2001)
IKNv2 (Jan. 2002)



Calculated:	7.93
IKNv2:	7.46
IKNv1:	6.94

vas_benmac_03

Conclusion

- Two algorithms for solving the bottleneck problem have been shown
- The result is the optimum solution to this problem
- Both algorithms scale well
- Thus, both can be used to determine the fair bottleneck rates
  - Theoretical benchmarking
  - On-line scheduling