Network Border Patrol: Preventing Congestion Collapse and Promoting Fairness in the Internet

Celio Albuquerque, Brett J. Vickers, Tatsuya Suda

Outline

- The Problem
- Existing Solutions
- Network Border Patrol
  - Feedback Control Algorithm
  - Rate Control Algorithm
- Simulations and Testing
- Conclusions

The Problem

- Congestion Collapse
  – Poor retransmission strategies
  – Rise of streaming video in the early 2000s
- Unfair bandwidth allocations
  – Differing TCP congestion algorithms
  – TCP ‘bias’ towards short RTT

Existing Solutions

- Logic in the Routers
  - Weighted Fair Queueing
  - Core-stateless Fair Queuing
  - CHOKe
- These are more complicated than FIFO
- They often do not work if your goal is *global* max-min fairness
  - If at router A, flows X and Y are allocated equally, and then only X encounters a later bottleneck, X will be overallocated at A.

Network Border Patrol

- Schematically similar to core-stateless, pushing flow classification and handling to the edge routers
- Categorize routers as ingress and egress routers
  - Note that a single router will act as both depending on which flows are being looked at
- Ingress routers separate packets into logical flows
- Egress routers measure the outbound capacity for each logical flow
- Ingress routers meter logical flows based on egress capacity

Network Border Patrol

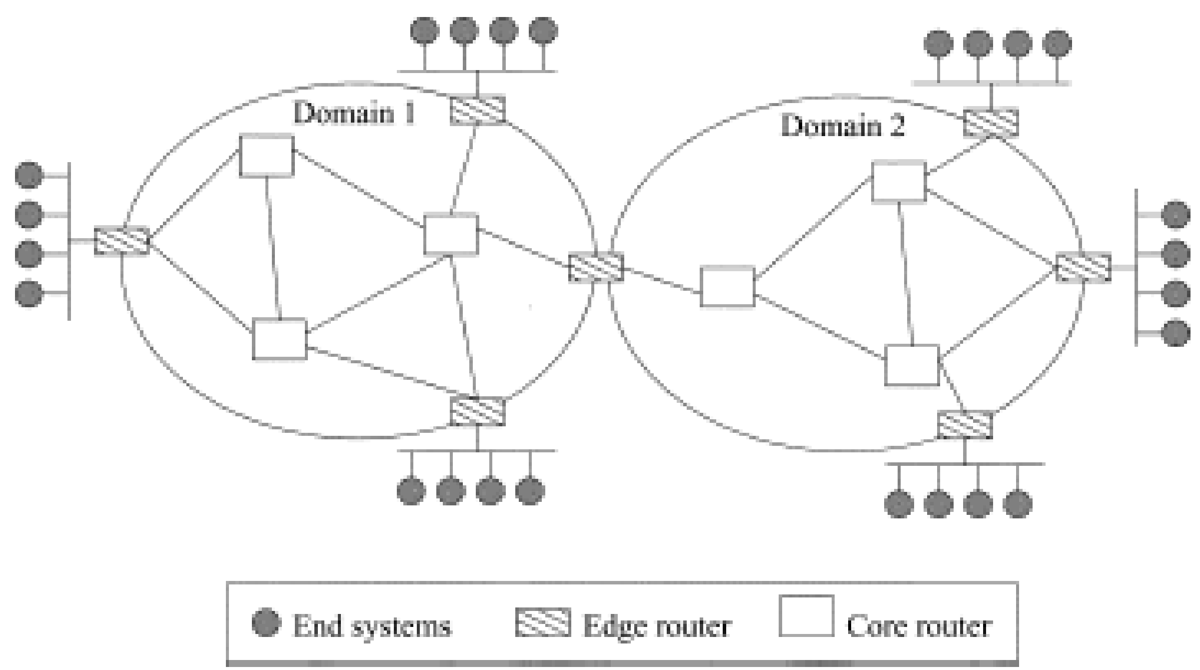


Fig. 2. Core-stateless Internet architecture assumed by NBP.

Feedback Control Algorithm

- Controls how the ingress and egress routers exchange packets
- A feedback packet is an ICMP packet (ping packet)
- In addition to exchanging flow data, they can be metered to sample internal congestion (through RTT)

Feedback Control Algorithm

- At ingress, a router categorizes a packet into a flow.
- The router increases the counter on that flow by n, where n is the size of the packet
- When the counter for a flow reaches T_x , create a "forward" packet
- A forward packet contains a timestamp and a list of unique identifiers for each of the N flows that the ingress router has seen for a given egress router

Feedback Control Algorithm

- At egress, a router generates a “backward” packet every time it receives a forward packet.
- A backward packet contains an associative array containing each flow and its outbound capacity.
- This packet is sent back to the ingress router and is used for traffic flow management (throttling, etc)

Network Border Patrol

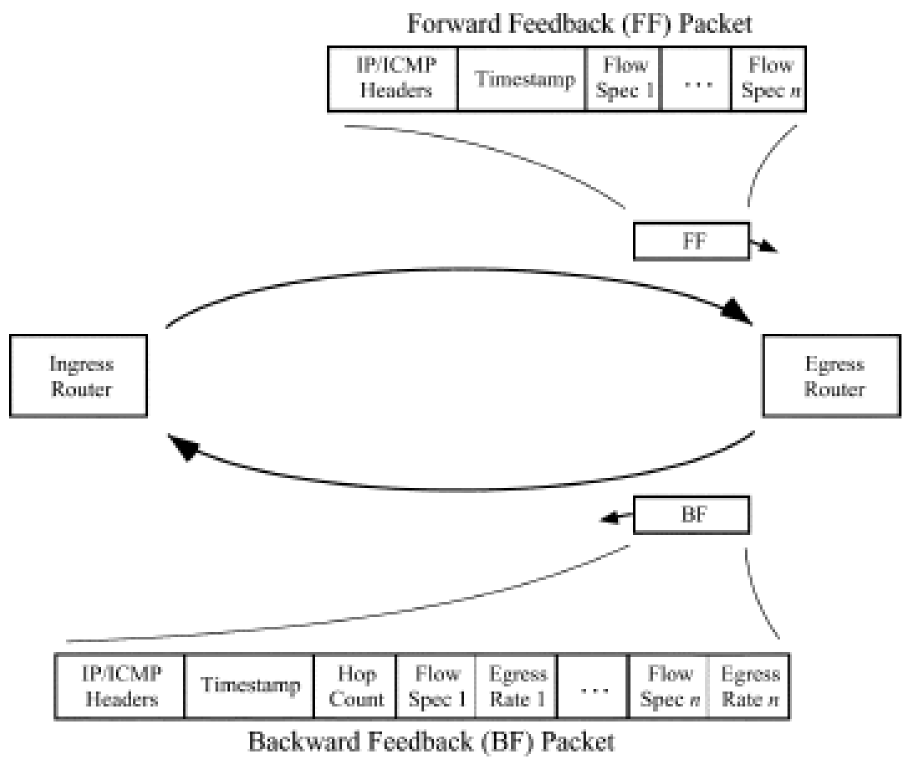


Fig. 5. Forward and backward feedback packets exchanged by edge routers.

Rate Control Algorithm

- Ingress routers use a Rate Control Algorithm to regulate the rate at which flows enter the network
- TCP-like implementation with two phases: slow start and congestion avoidance
- Track the RTT of the feedback packets and use the current RTT and best observed RTT in the algorithm

Rate Control Algorithm

```
on arrival of Backward Feedback packet p from egress router e
    currentRTT = currentTime - p.timestamp;
    if (currentRTT < e.baseRTT)
        e.baseRTT = currentRTT;
    deltaRTT = currentRTT - e.baseRTT;
    RTTsElapsed = (currentTime - e.lastFeedbackTime) / currentRTT;
    e.lastFeedbackTime = currentTime;
    for each flow f listed in p
        rateQuantum = min (MSS / currentRTT, f.egressRate / QF);
        if (f.phase == SLOW_START)
            if (deltaRTT × f.ingressRate < MSS × e.hopcount)
                f.ingressRate = f.ingressRate × 2 ^ RTTsElapsed;
            else
                f.phase = CONGESTION_AVOIDANCE;
        if (f.phase == CONGESTION_AVOIDANCE)
            if (deltaRTT × f.ingressRate < MSS × e.hopcount)
                f.ingressRate = f.ingressRate + rateQuantum × RTTsElapsed;
            else
                f.ingressRate = f.egressRate - rateQuantum;
```

Fig. 6. Pseudocode for ingress router rate-control algorithm.

Fairness?

- NBP by itself is not “fair”, it only meters a flow based on the share it can claim of its smallest bottleneck.
- Thus if flows are competing for a bottleneck, they may still be treated unfairly.
- Introduce a fair queueing mechanism to NBP, such as CSFQ or rainbow fair queueing.

Enhanced CSFQ

- CSFQ cannot be easily plugged into NBP because CSFQ does not preserve the delay characteristics of true fair queueing, because it does not separate flow buffers.
- This can cause problems with congestion schemes that rely on RTT to throttle without packet drops.
- Instead of a single buffer, E-CSFQ uses a second, high priority buffer.
- This buffer is serviced first, and is used by flows using less than their “fair” share.

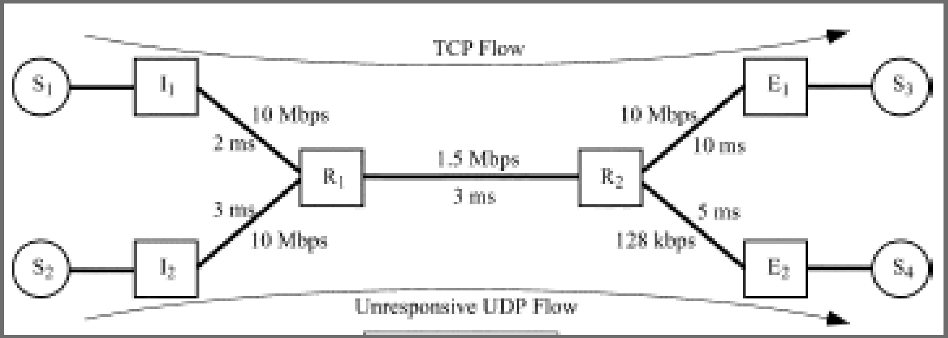
Enhanced CSFQ

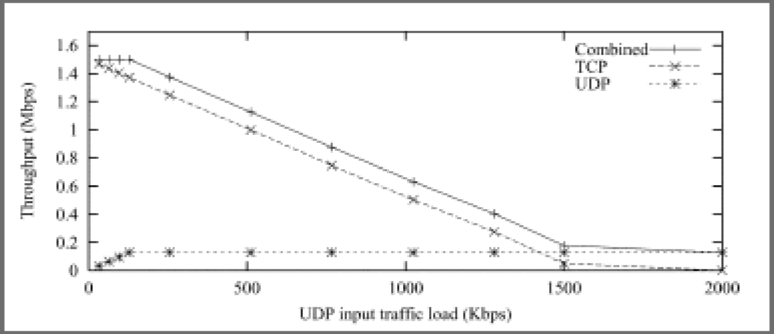
- The addition of a second buffer may cause packet reordering issues, but the writers assert this should be rare, because it requires a flow to be ‘recategorized’ from low to high.
- The writers say these situations should be unlikely, because it requires a flow to drastically change its flow rate, or for bandwidth to appear. This will inherently mitigate reordering because it allows the low-priority queue to be serviced more quickly.

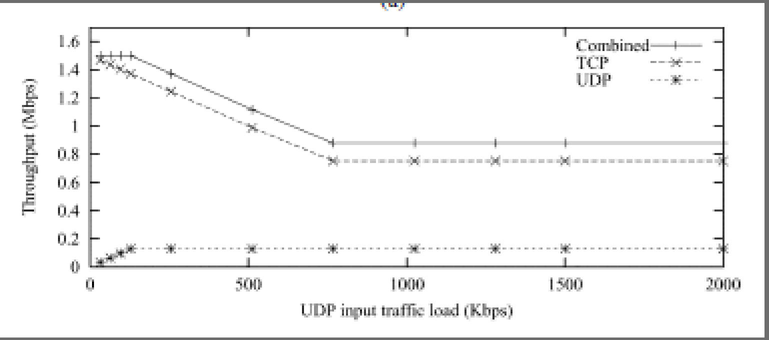
Simulations

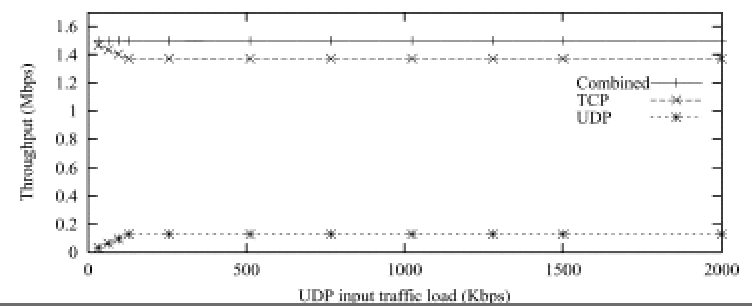
- Implemented in ns-2
- Experiment one deals with the ability of NBP to prevent congestion collapse
- Experement two deals with the ability of ECSFQ to provide fair allocations
- Experiment three deals with scalability of NBP
- Experiments were run for 100s

Congestion Collapse – Single Link



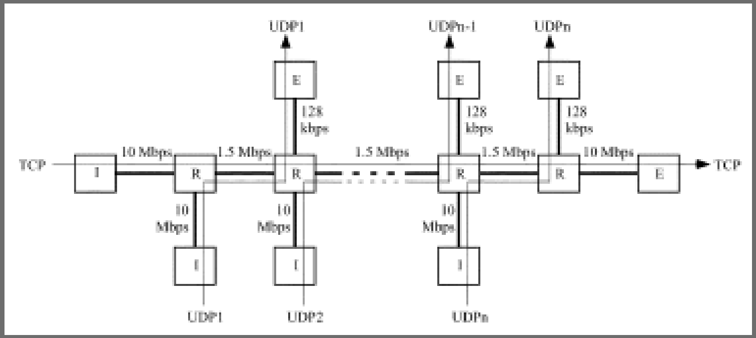


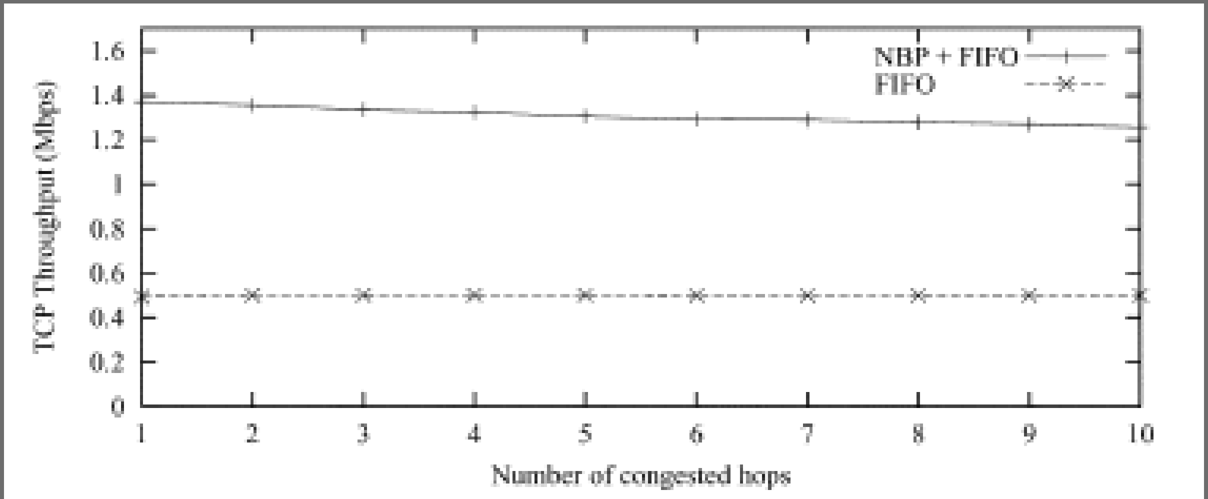




Charts from Albuquerque et al

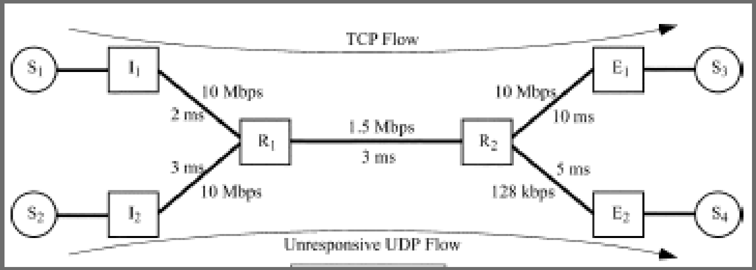
Congestion Collapse – Multi Links

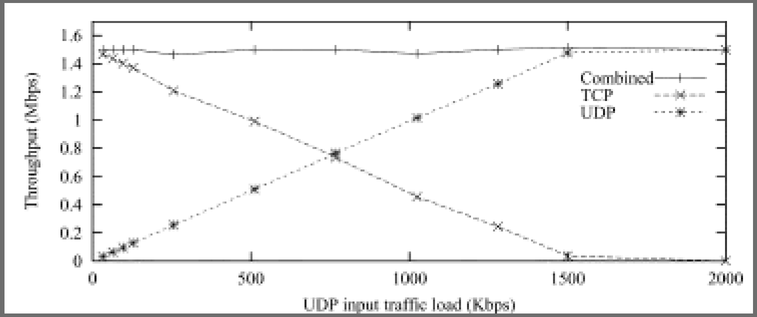


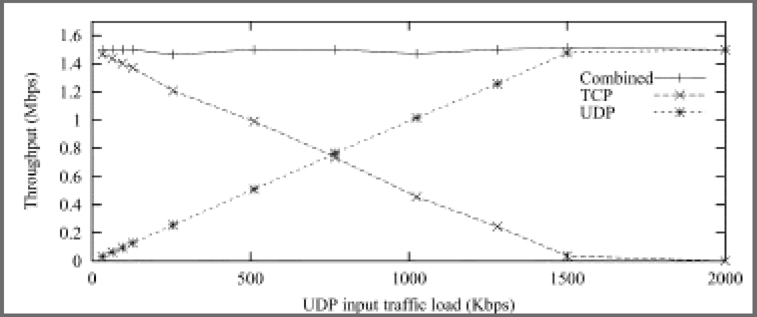


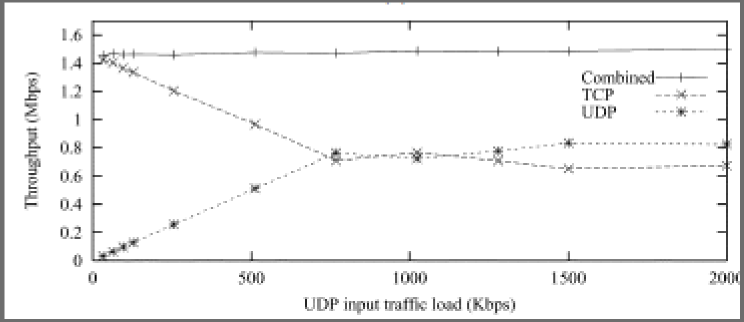
Charts from Albuquerque et al

Fairness – Single Link









 Charts from Albuquerque et al

Fairness – Multi Link

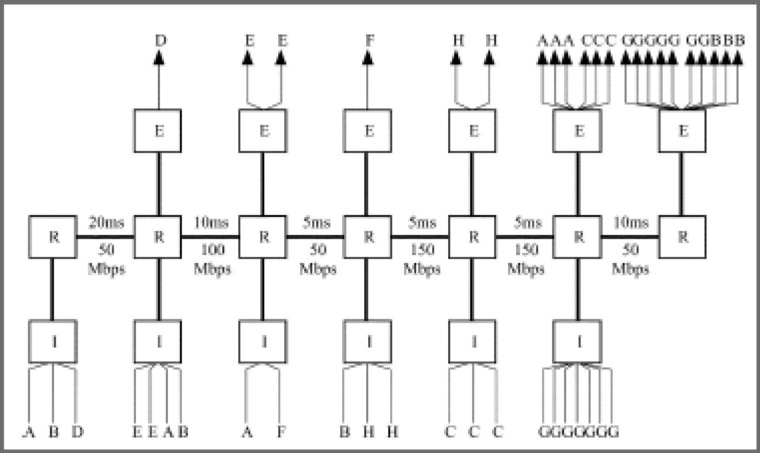


TABLE II
PER-FLOW THROUGHPUT IN THE GFC-2 NETWORK

Flow Group	Ideal global max-min fair share (Mbps)	Simulation results: Throughput using FQ only (Mbps)	Throughput using NBP with FIFO (Mbps)	Throughput using NBP with WFQ (Mbps)	Throughput using NBP with ECSFQ (Mbps)
A	10	8.32	10.96	10.00	10.40
B	5	5.04	1.84	5.04	4.48
C	35	27.12	31.28	34.23	31.52
D	35	16.64	33.84	34.95	32.88
E	35	16.64	37.76	34.87	33.36
F	10	8.32	7.60	10.08	8.08
G	5	4.96	1.04	4.96	5.28
H	52.5	36.15	46.87	50.47	47.76

Charts from Albuquerque et al

Scalability – Multiple Flows

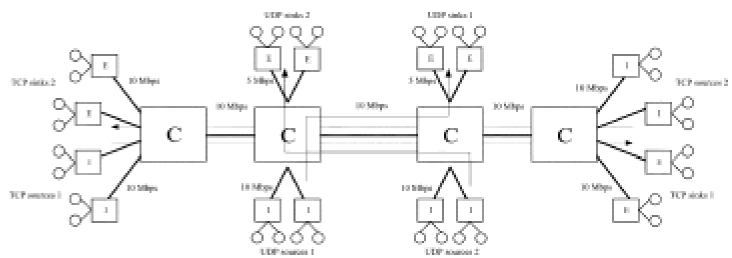
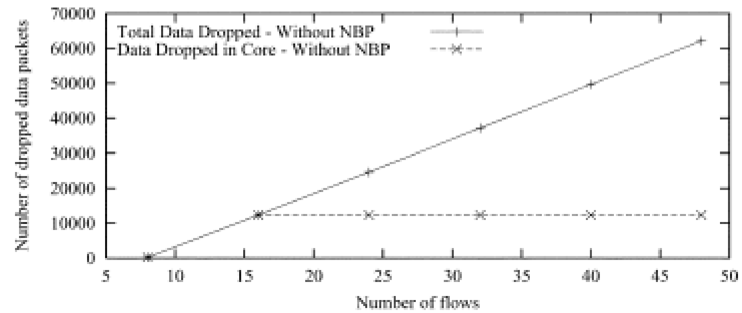
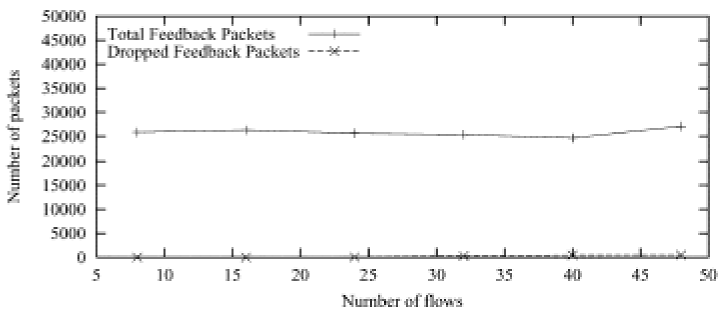
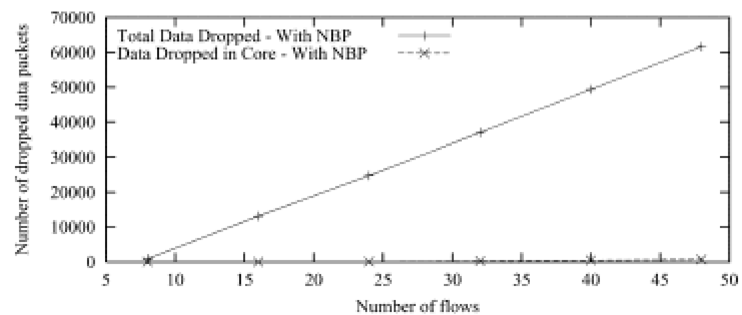


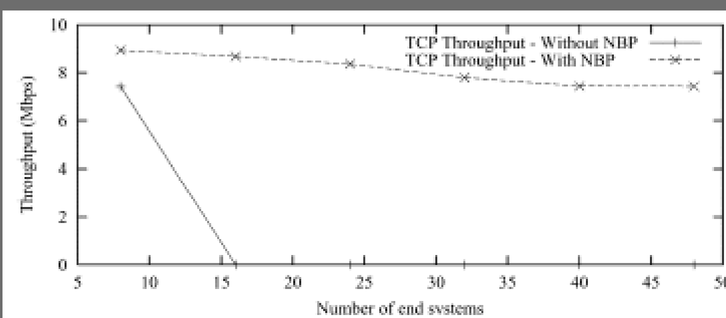
Fig. 16. Simulation model for evaluating scalability.



(a)







Charts from Albuquerque et al

Scalability – Crossing Flows

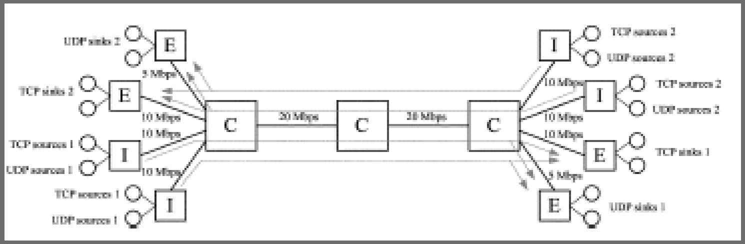


TABLE III
PERFORMANCE RESULTS FOR INGRESS ROUTERS COMMUNICATING WITH MULTIPLE EGRESS ROUTERS

UDP load (Mbps)	TCP throughput (Mbps)	UDP throughput (Mbps)	Number of feedback packets	Number of dropped packets		
				Feedback packets	Data packets	In the Core
2	9.99	2.00	1642	0	25	0
4	9.98	3.99	2071	0	171	0
6	9.98	4.92	2234	0	1230	0
8	9.98	4.98	2260	0	3660	0
10	9.98	4.99	2303	0	6163	0
12	9.97	4.99	2332	0	8780	0
16	9.97	4.99	2327	0	13756	0
20	9.94	4.99	2315	0	18615	0

Charts from Albuquerque et al

Raised Issues, Future Work

- Flow classification overhead may become a concern, perhaps a coarser flow classification to reduce the number of macro-flows could be used.
- Scalability problems may be further reduced by incorporating “trust”. “Trust” other subnets and accept their edge information, too.
- NBP must be deployed over an entire edge at once.
- Multicast greatly complicates the situation by breaking the “one flow->one egress” assumption.

Conclusion

- This paper presented a possible solution to the problem of ‘congestion collapse’, the fact that fair queuing algorithms can only ‘look-backward’
- NBP uses the idea of a circular communication in router coordination, rather than the one-way from ingress to core of CSFQ.
- They included a large amount of experimental data to demonstrate their point.
- Has scalability been fully addressed?