# Internet Border Patrol

CS-03 Dhruv Gajjar, CS-06 Rudransh Jani, CS-17 Dhwani Patel

This project will see you working on a system that declutters undesirable traffic on a website (example). The system will look for unresponsive packets in the network and remove them. These packets can clog the network at times and make it difficult for people to browse later. This approach called “Internet Border Patrol” helps to control any type of network congestions. The project would involve the creation of the system using scripted language and base Python programming that provides conditions to check for unresponsive packets in the network. Unresponsive packets can be utilized by malicious users who may fill in viruses and worms that can attack, compromise, and/or destroy the entire system. Hence, this project/technique helps in declogging the network and also ensuring that there are no stray packets that may be misused.

We will be using Python as a scripting language.

# Outline

- Problem
- Current Solutions
- Internet Border Patrol

# Problems

- Congestion Collapse
- Unfair Bandwidth Allocation

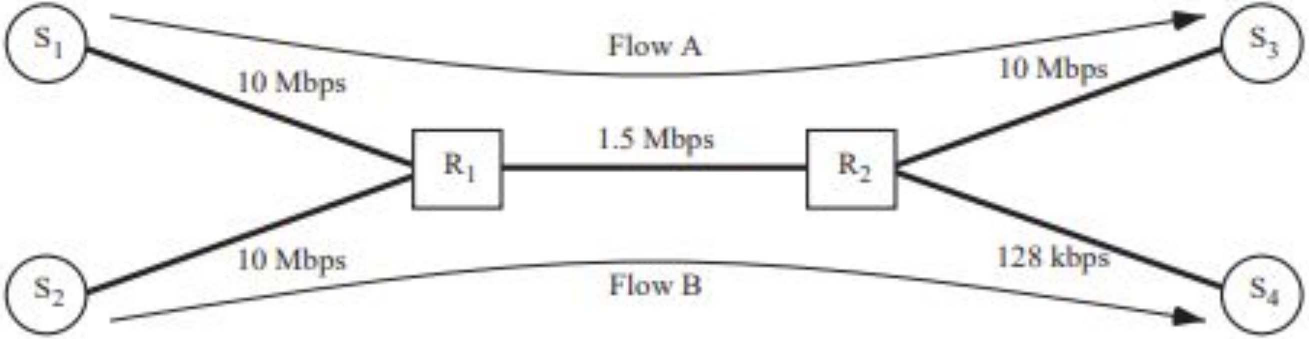
# Congestion Collapse

Congestion collapse from undelivered packets—arises when bandwidth is continuously consumed by packets that are dropped before reaching their ultimate destinations . Unresponsive flows,which are becoming increasingly prevalent in the Internet as network applications using audio and video become more popular,are the primary cause of this type of congestion collapse, and the Internet currently has no way of effectively regulating them.

# Unfair Bandwidth Allocation

Unfair bandwidth allocation—arises in the Internet for a variety of reasons, one of which is the presence of unresponsive flows. Adaptive flows (e.g., TCP flows) that respond to congestion by rapidly reducing their transmission rates are likely to receive unfairly small bandwidth allocations when competing with unresponsive or malicious flows. The Internet protocols themselves also introduce unfairness.

# Example


$S_1$
$S_2$
$S_3$
$S_4$
Flow A
Flow B
10 Mbps
10 Mbps
10 Mbps
1.5 Mbps
128 kbps
$R_1$
$R_2$

# Congestion Collapse

At the first bottleneck link (R1-R2), fair queueing ensures that each flow receives half of the link’s available bandwidth (10 Mbps). On the second bottleneck link (R2-S4), much of the traffic from flow B is discarded due to the link’s limited capacity (128 kbps). Hence, flow A achieves a throughput of 10 Mbps and flow B achieves a throughput of 128 kbps. Clearly, congestion collapse has occurred, because flow B packets, which are ultimately discarded on the second bottleneck link, unnecessarily limit the throughput of flow A across the first bottleneck link.

# Unfair Bandwidth Allocation

while both flows receive equal bandwidth allocations on the first bottleneck link, their allocations are not globally max-min fair. A globally max-min fair allocation of bandwidth would have been 13.72 Mbps for flow A and 128 kbps for flow B.

# Current Solutions

- Logic in the Routers
  - Weighted Fair Queueing
  - Core-stateless Fair Queuing

- 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.

# Weighted fair queueing

Weighted fair queueing (WFQ) is a method of automatically smoothing out the flow of data in packet-switched

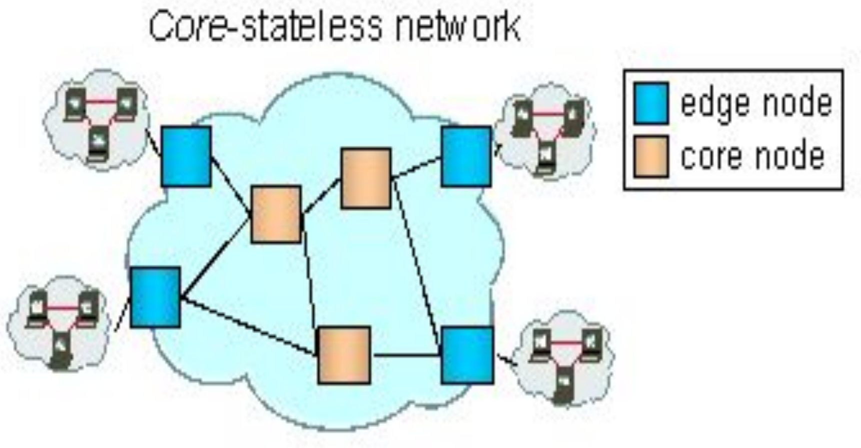
communication networks by sorting packets to minimize the average latency and prevent exaggerated discrepancies

between the transmission efficiency afforded to narrowband versus broadband signals.

In WFQ, the priority given to network traffic is inversely proportional to the signal bandwidth.

Thus, narrowband signals are passed along first, and broadband signals are buffered.

# Core-stateless Fair Queuing



# Core-stateless Fair Queuing

In CSFQ each edge node

- **estimate** the incoming rate of each flow, and use it to label flow's packets

In turn all nodes, including edge nodes, perform the following operations

- periodically **estimate** the fair rate, *f* along the outgoing link.
- upon packet arrival **compute** the forwarding probability *P* based on the packet label (which represents the current value of the estimated flow rate) the fair rate of the output link. Then **forward** the packet with probability *P*
- when a packet is forwarded **replace** its label with the minimum between its previous value and the fair rate of the output link

# Core-stateless Fair Queuing

**Flow Rate Estimation** Edge nodes estimate the rate of each flow based on exponential averaging.

**Link Fair Rate Estimation** When the link is congested, the fair rate *f* is computed such that the rate of the aggregate forwarded rate equals the link capacity. Since the aggregate forwarded rate is a non-decreasing and monotonic function of *f* we use an iterative algorithm based on linear interpolation to compute it.

When the link is uncongested we take *f* to be the maximum among the arrival rates of the incoming flows (i.e., the largest label of a packet seen during a certain period).

A link is assumed to be congested/ucongested if during an predefined time interval the aggregate forwarded rate is always larger/lower than the link capacity.

# Core-stateless Fair Queuing

**Computing Packet Forwarding Rate** Upon a packet arrival each node computes its forwarding probability *P* based on the following formula

p=min(1,f/r)

where *r* is the current estimated rate of the flow, which is contained in the packet label, and *f* is the fair rate of the output link. By forwarding the packet with probability *P*, the expected rate of the flow's forwarded traffic is

r’=r*min(1,f/r)=min(f,r)

# Internet/Network Border Patrol

The basic principle of NBP is to compare, at the borders of the network, the rates at which each flow’s packets are entering and leaving the network. If packets are entering the network faster than they are leaving it, then the network is very likely to be buffering or, discarding the flow’s packets. In other words, the network is receiving more packets than it can handle. NBP prevents this scenario by “patrolling” the network’s borders, ensuring that packets do not enter the network at a rate greater than they are able to leave it. This has the beneficial effect of preventing congestion collapse from undelivered packets, because an unresponsive flow’s otherwise undeliverable packets never enter the network in the first place.

# Internet/Network Border Patrol

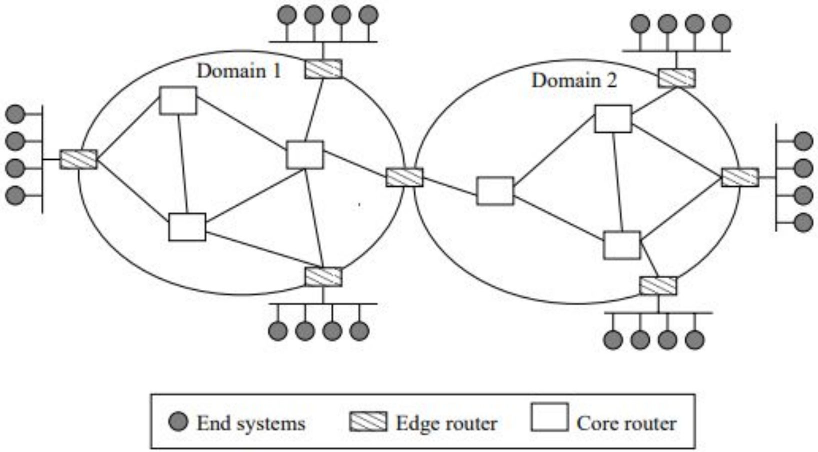


Fig. 2. The core-stateless Internet architecture assumed by NBP

# Internet/Network Border Patrol

- Expected to monitor and control the rates of individual flows
- Added Communication overhead, since in order for an edge router to know the rate at which its packets are leaving the network, it must exchange feedback with other edge routers.

# Edge Router

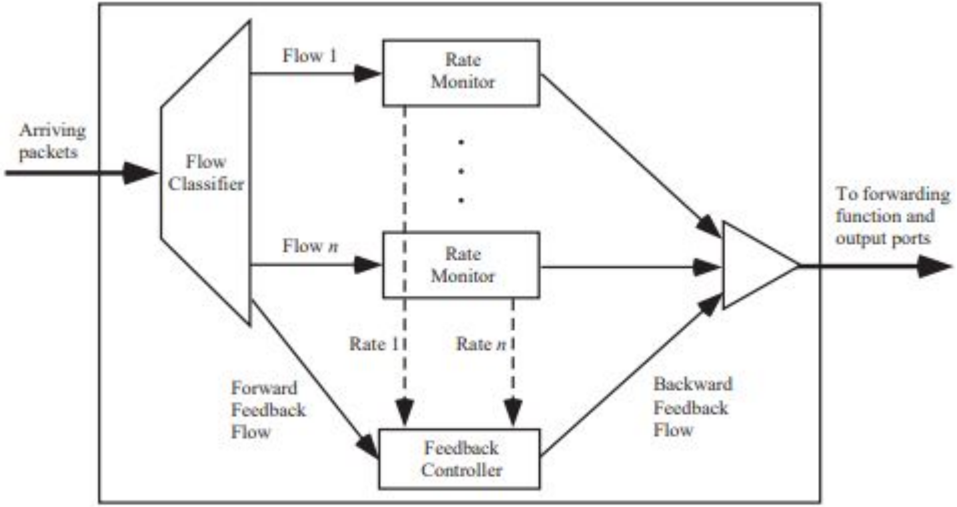
1. Ingress Router

   An Edge Router Operating on the Flow passing into a network is called an Ingress Router.

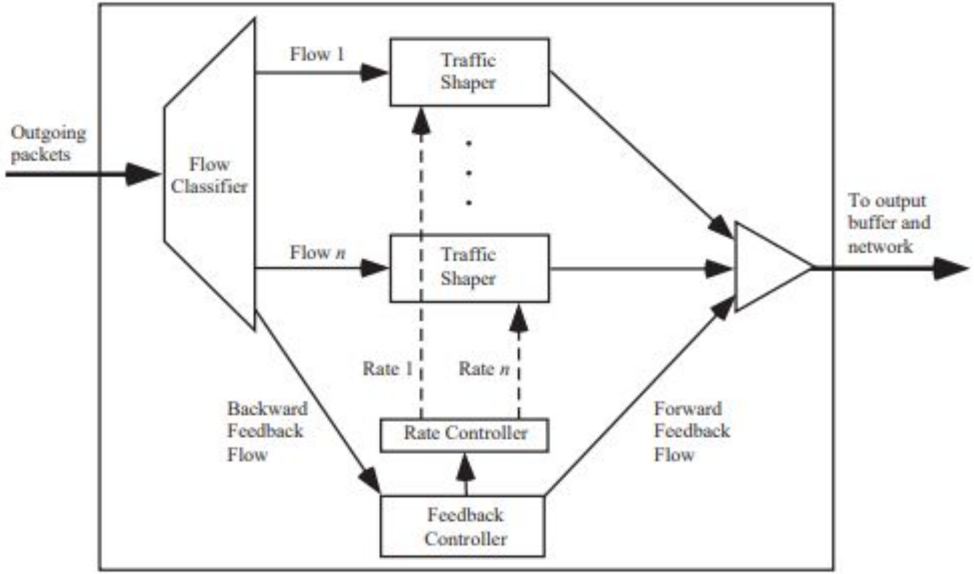
2. Egress Router

   An Edge Router Operating on the Flow passing outside a network is called an Egress Router.

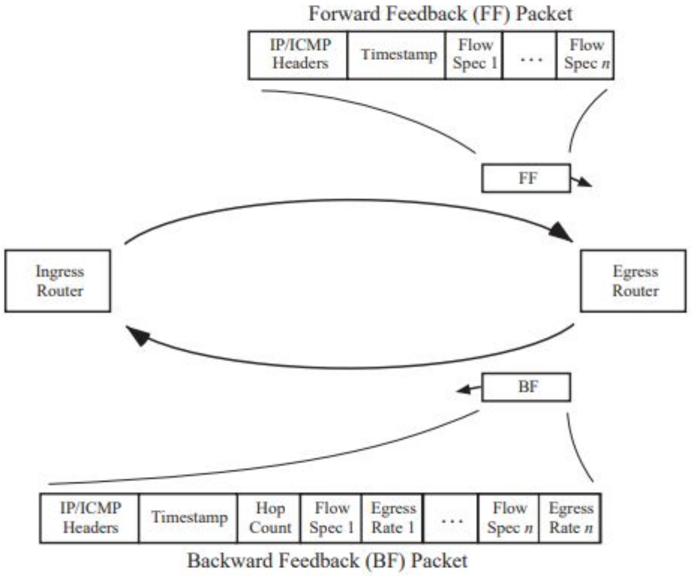
# Ingress Router



# Egress Router



# Packets Exchanged By the Edge Router



Forward Feedback (FF) Packet

| IP/ICMP Headers | Timestamp | Flow Spec 1 | . . . | Flow Spec *n* |
|---|---|---|---|---|

| IP/ICMP Headers | Timestamp | Hop Count | Flow Spec 1 | Egress Rate 1 | . . . | Flow Spec *n* | Egress Rate *n* |
|---|---|---|---|---|---|---|---|

Backward Feedback (BF) Packet