Modeling of the Linux Switching Architecture

V. GUFFENS, G. BASTIN

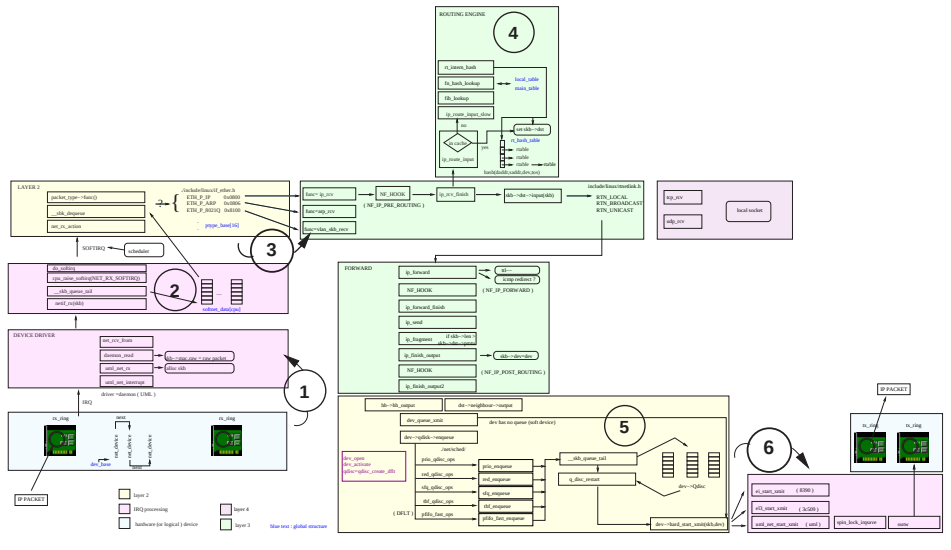
Center for Systems Engineering and Applied Mechanics, UCL

MOTIVATION

The use of system analysis and feedback control theory in networking application is receiving increasing attention by the scientific community.
The use of hardware independent protocols such as TCP/IP has made possible the interconnection of heterogeneous physical networks and its wide deployment has lead to a huge number of applications requiring different services from the networks such as guaranteed bandwidth or bounded delay (Quality of Service or QoS).
Unfortunately, the IP protocol is intrinsically best effort and the end-to-end control brought by the use of a connection oriented protocol such as TCP is not sufficient to provide the required QoS. We believe that the limitation due to the best effort nature of IP can be alleviated by controlling the traffic.
The first step in designing new traffic control strategies is to establish a dynamical model of the network.
The purpose of this poster is to show the modeling of a typical switching platform, namely the 2.4 Linux kernel, modeling base on the queueing theory and on the qualitative analysis of the Linux switching architecture.

1 Construction of the model

1.1 The path of a packet in the Linux kernel [4] [1] [3]



1 The packet arrives from the network. It is placed in some hardware memory on the NIC. The card issues a **hardware interrupt**. The processor executes the device driver and copy the packet from the card to the main memory into a structure called **sk_buff**. The packet can be copied byte per byte or using Direct memory access.

2 This **sk_buff** is queued in a FIFO manner in the input queue called **softnet**. There exists one such input queue per processor to avoid any locking issue. If the queue is full (300), the queue is **throttled** and the packet is discarded. The queue will be throttled until the queue length reach 0.

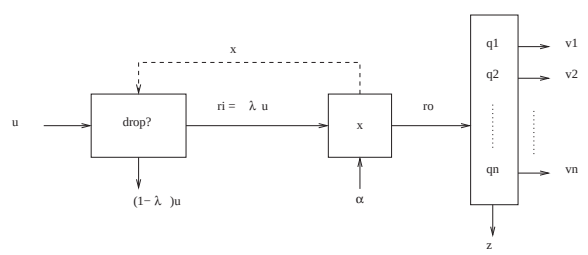
3 Before returning from the interrupt, the driver issues a **soft interrupt**. The scheduler is invoked. Once the software interrupt is allowed to execute, the **sk_buff** is dequeued and the ISO layer 2 information is analyzed. According to the value of the **type** field, the packet is handed to the correct layer 3 function.

4 The destination IP address is extracted and the **route cache** is inspected. If the entry is not present in cache, a look up is performed in the main routing table. The **next hop** information is recorded in the **sk_buf**. If there is no entry in the routing table, the packet is discarded (black hole).

5 According to the routing decision, the **sk_buff** is then queued to the correct outgoing interface. There exists one output queue per interface and different queueing methods can be applied such as **RED**, **TBF**, **WFQ**.

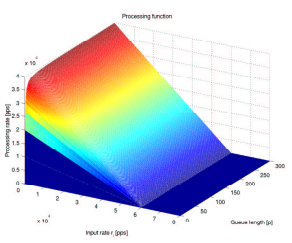
6 The packet is transfered to the hardware memory and the card is instructed to send the packet on the network.

1.2 A simple fluid flow mathematical model



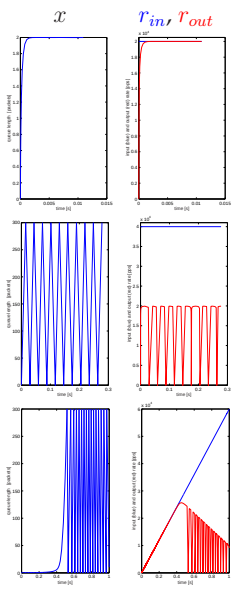
1.2.1 Non-linear system [2]

$$\begin{aligned} f(u,x) &= \mu.\frac{x}{x+1}.\max\{(1-\frac{u}{\beta}),0\}.p(\alpha) \\ \dot{x} &= r_i - f(u,x) \end{aligned}$$



u, x	input rate and input queue length
μ	maximun rate of packets switching achieved by the processor if it was not interrupted
$\frac{1}{\beta}$	time spent in interrupt to process one incoming packet (or to drop it)
$p(\alpha)$	function of the processor ocupation(set to $p(\alpha) = 1$
$\lambda \in \{0,1\}$	dropping indicator All the packets are dropped when the queue is full until the queue length reach zero again
q_i, v_i	output queue and output rate for the ith interface

2 Simulation



The system is stable

$\mu = 40000$ and $\beta = 80000$. The input function is a step of value $20000[pps]$. We can clearly see the output rate growing towards the input rate while the input queue is filling up. The system is stable, we don't loose any packet and the queue length stabilizes at its equilibrium point.

The system is NOT stable

$r_{in} = 40000[pps]$. This is more than the system can handle. The system will converge to its maximum switching rate which is 20000 for this input rate. The queue is filling up until its maximum value and is throttled until it reaches 0 again.

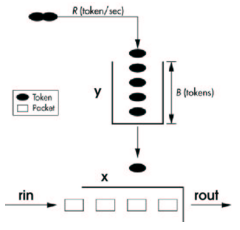
From stability to instability

The input is now set to a ramp. This is giving us an idea of the input-output steady state relationship. This curve is observed in practice.

3 The output queue

3.1 The output queue is a Token Bucket Filter [5]

- tbf = traffic shaper
- sending one packet = taking a token from the bucket
- We put R tokens per second in the bucket
- The bucket is initialized with b tokens
- Everything is re-initialized every Δt



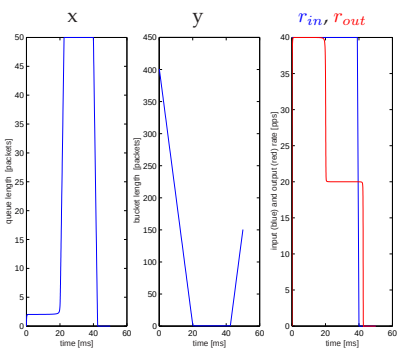
3.2 An equivalent fluid-flow model

$$\begin{aligned} r_{out} &= \mu.\frac{x}{x+1}\frac{y}{y+1} \\ \dot{x} &= r_{in} - r_{out} \\ \dot{y} &= R - r_{out}, \quad y(0) = b \end{aligned}$$

x : output queue length, y : number of tokens in the bucket,(R, b) : parameter of the tbf.

3.2 Simulation of the tbf

- $R = 20000$,$b = 400$, $r_{in} = 40000$ [pps] during 40 [ms]
- $r_{out} = 40000$ [pps] until $y = 0$ (after 20 [ms])
- output rate is limited to R
- output buffer accumulates the difference until saturation



References

[1] Daniel P. Bovet and Marco Cesati. *Linux Kernel*. O'Reilly, 2001.

[2] H. Mounier G. Bastin. Compartmental modelling for traffic control in communication networks, submitted to ieee transactions on automatic control, 2002.

[3] Paul Gortmaker. Linux ethernet-howto, 2000.

[4] Alessandro Rubini and Jonathan Corbet. *Linux Device Drivers*. O'Reilly, 2001.

[5] Andrew Tanenbaum. *Réseaux, cours et exercices*. Dunod, 1996.