# Project

Due March $27^{th}$ 2009

Switching

## The problem

The project deals with one of the most important tasks performed inside a DLL switch: forwarding frames. When a frame arrives to a switch, it needs to be sent forward, closer to its final destination. A switching table (a form of routing table) helps in deciding which output link is to be used.

The switch is made of 7 NIC cards, each controlling one pair of unidirectional links.

The lines are point–to–point (the NIC is at one end; another identical NIC is at the other end which is located at an unknown but small distance).

From the point of view of this project, the basic components of each card are 2 buffers (input and output) and a **link manager**. The manager moves frames (which must be correct if they reached this stage) from its input buffer to the output buffer of one of the other 6 NIC cards.

The cards are connected by a central connector which (for the purpose of this project) can have one of three architectures:

**Broadcast bus:** a single shared link connects all the cards. A card that wants to transfer a frame from its buffer to the buffer of another card has to reserve the bus first.

**Banyan switch:** there is a unique path between each pair of cards but some points along this path are shared with other paths (and may become bottlenecks).

**Crossbar switch:** Each card has its own receive bus and its own transmit bus. Each receive bus is connected to every transmit bus. Contention arises when two cards want to send frames to the same receiver.

The three architectures are well covered in the literature. If you uncover variations in their design, you can choose the variation you like best. Each architecture has the same data rate: 8 times the data rate of a link (if the link has a 1 Gb/s rate, the inter–connection will be able to move data at the rate of 8 Gb/s in each of its channels).

## The single buffer problem

A NIC has a single input buffer. When the card is trying to move its contents to the output buffer of another NIC, it must reject anything arriving in its link (there is no space to store it).

That implies that any frame so rejected will have to be retransmitted later.

A NIC has one output buffer. When another card is trying to overwrite a full output buffer, it must be prevented from doing so until the buffer is emptied.

You must design a way that will allow your switch to function without ever dropping any frames. So, once a frame makes its way to a NIC and is accepted, it must be forwarded successfully.

So where are frames dropped? The original sender (not located within this switch, but at the other end of one of the 7 links connected to the switch) has limited buffer space. Let this space be equal to 2 buffers (of one frame each). If a third frame is submitted to it when the two buffers are full, it is brutally dropped (does count towards throughput, but not effective throughput).

## Your task

Your task is to design the software that controls the movement of frames from output buffers (receiver's) to input buffers (transmitter's).

Ideally, your work should consist of the following:

- Choose one of the three inter–connection architectures.
- Design the reservation algorithm for the inter–connection. If there is a possibility of contention (there will be), design a collision resolution algorithm that you will use.
- Design a model of frame arrivals to the remote ends of the links connected to the switch. Note that this model is unrelated to what the capacity of the switch is: it must allow for overloading the system (in which case the effective throughput will be smaller than the offered load).

  Use this model to obtain frame arrivals at the NIC cards (do not forget that some frames will get rejected). This can be drawn from your second assignment or from a probabilistic model of your own design (which must be adequately complete). Add a randomised decision algorithm that chooses where to forward a frame.
- Implement a simulator for the inter–connection of your choice. The simulator will include both the reservation+collision algorithm and the actual bit movement inside the inter–connection (no clock drifting and no errors).

- Model the operation of emptying output buffers (the same as transmitting frames, but without clock problems and error–free).
- Combine your models (NIC and inter–connection) and use the resulting simulator to determine how your switch handles traffic. The parameters to be tracked are:

  **Delay inside the switch:** the amount of time elapsing between the moment a frame is accepted by the NIC manager and the moment it leaves an output buffer.

  **Effective throughput:** the proportion of the frames that made it to the output buffers as a function of the rate of frame arrivals to the remote ends of the links connected to the switch.

## Project requirements

This is an individual project. You may, however, team up with another student to produce a joint project provided that:

- You will investigate 2 different inter–connection architectures. The project will contain a section that will compare the two architectures; the comparison will include price/performance considerations.
- The frame arrival model is realistic enough (this will be negotiated).
- The same grade will be given to both students.

Each student will meet with me 3 times before the project is submitted. Each meeting will include:

- A written progress report prepared before the meeting.
- A written plan of future work (also prepared before the meeting).
- A draft project report to the extent that it exists. This should be prepared at least one day before the meeting.

Team members may choose to combine their meetings.

A presentation will be necessary to get credit for the assignment.