2030 NETWORK

Synchronous
Symbolic
Hierarchy

Balyberdin A.L.
Rutel.Nsk@Gmail.com

НСКФ
НАЦИОНАЛЬНЫЙ СУПЕРКОМПЬЮТЕРНЫЙ ФОРУМ

Copyright

All ideas and algorithms , described in this article are the result of my independent and entirely self-sufficient intellectual activity.

As the author, I give permission to freely use, modify and supplement all ideas and algorithms to any person or organization in any type of projects, subject to the obligatory reference to my authorship.

 Rutel.Nsk@Gmail.com

Issues to Solve

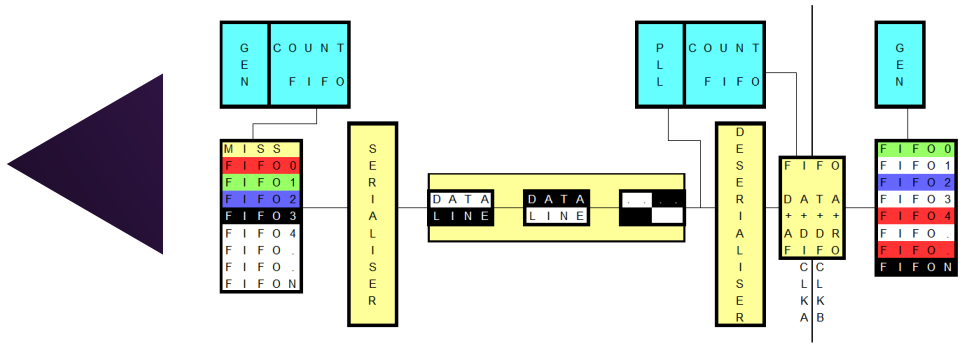
01 In 2018, the 13th ITU-T Study Group (SG-13) established the Network 2030 technology focus group (FG NET-2030); the group studied the possibilities of fixed networks for the period up to 2030. It was assumed that Network 2030 could be built on a new or improved network architecture that could evolve from the modern networks or differ greatly from it.

02 Please consider and take the results of my work as a basis for the promising networks. I propose a circuit-switching network with the working title “Synchronous Symbolic Hierarchy” (SSH). SSH network fully complies with all the basic requirements formulated for the promising networks of “Network 2030”.

03 The new network architecture allows a user to replace all existing networks as well as various local interfaces transparently (including low-level ones). This document contains only the telecommunication component of the project, but there is also a computing component (DataFlow computing systems).

Typical Link Structure

Almost all high-speed connections are built on the unified principle. Synchronization block, FIFO buffer selection block with data for transmission, Line encoding and Serialization block, Communication cable, Clock frequency recovery block, Deserialization block, Evaluation unit for the received data record address , FIFO block for crossing the boundary with different clock frequencies, FIFO block for storing the received data



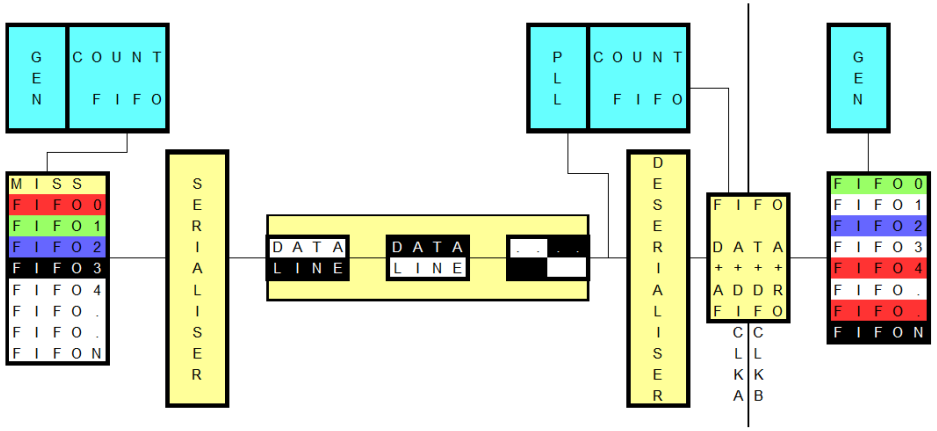
Data is transmitted synchronously, which makes such a link suitable for use in the promising Network 2030.

Data is transmitted in chunks, called symbols. The symbol size in bits is determined by the ratio of the data transmission rate in a channel and clock frequency of the switch. In addition, a symbol includes various line encoding data, correction and service codes.

The “Slippage" Problem

The advantages of synchronous systems for transmission and switching are rapid data transmission rates and small buffer sizes for intermediate storage of the transmitted data.

But there are also disadvantages, one of them is the slipping effect. This effect appears at the boundaries of zones with different clock frequencies. It is impossible to make completely identical clock frequency oscillators; sooner or later they will start scattering. For the transmitted data, this will result in the overflow or underrun of the intermediate buffers.



To solve this problem, I propose to use the following algorithm: The clock frequency of the receiver and transmitter are approximately equal (plesiochronous). Among the transmitted symbols, we select one bit combination and assign “no data” property to it. When generating a stream of symbols, a transmitter automatically and evenly inserts such a symbol for every two thousand regular ones, which compensates for the inaccuracy of the oscillators up to +-250ppm. On the receiving side, this symbol is simply discarded.

You can enter a “transmission error” symbol and insert it if there are errors in decoding a symbol on the receiving side; this method will allow you to detect a specific symbol containing an error.

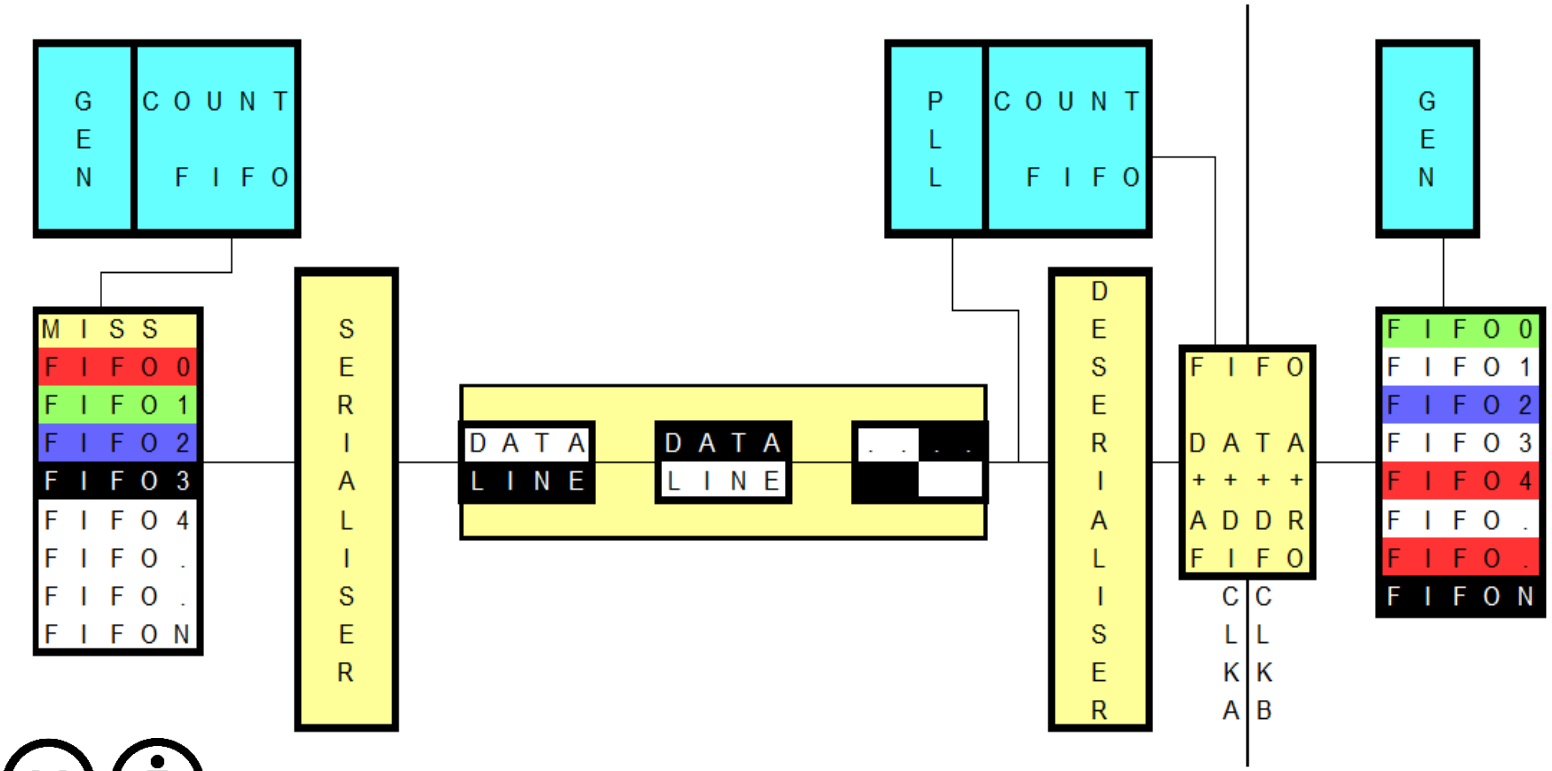
The Advantages of Slippage Equalization

The advantages obtained when using “no data” symbol. Adding this symbol ensures that on the receiving side (also for intermediate switches), a situation will not happen in which the receive buffer overflows.

Yes, there is a decrease in the efficiency factor of using a channel bandwidth, but 0.05% is not a value that is worth “worrying about”.

Besides, there will never be more than 1-2 symbols in intermediate buffers; therefore, the buffer size is guaranteed to be the same - 1-2 symbols (for any number of the intermediate switches).

The average number of data in the intermediate buffer will be 0.5 symbols, which guarantees the average switching time at the level of 50% of the transmission period (but not more than 2 periods).



Creation of Virtual Channels

The required number of concurrent communication sessions during the equipment operation can exceed trillions, which is many times greater than the number of physical links. A mechanism is needed to share one physical channel among a great number of processes.

Let's formulate the requirements for such a system:

- Maintaining the properties achieved at the stage of solving the slippage problem
- Instant channel creation without prior notification to a receiver
- No effect of the virtual channel creation errors on both already created and subsequent created channels
- Creation of a virtual channel with an arbitrary and precisely determined rate, both with its constant and asynchronous part
- No cross effect of already created channels

Physical Channel Division Algorithm

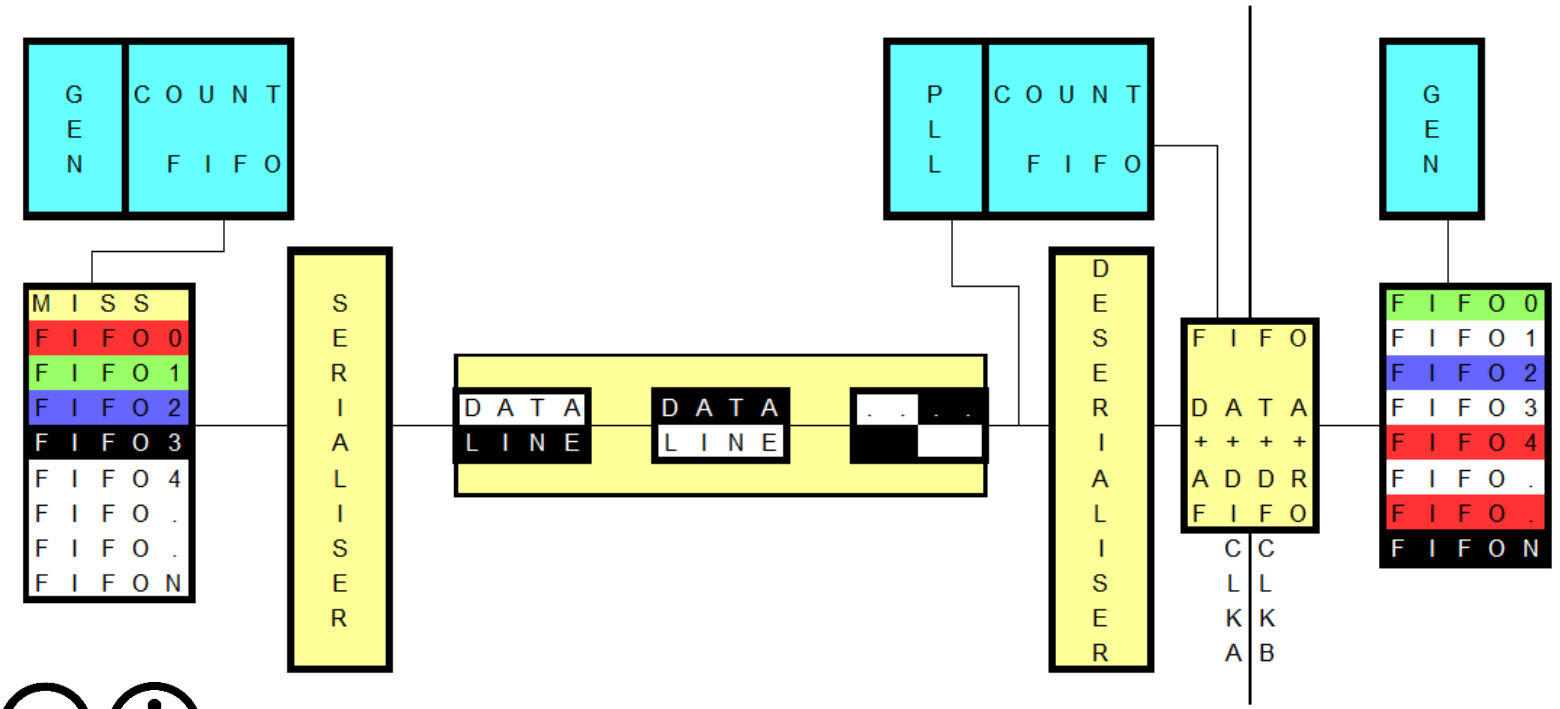
Obviously that a shared physical channel must have some structure, and a synchronized pointer for the receiver and transmitter to the current position within this structure is needed.

You can use a circular counter of the transmitted symbols as a pointer.

For synchronization, "no data" symbols can be used; the symbol size is large enough, and some of the bits can be used to transmit the counter value.

At the time of transmission, the counter value on the transmitting side is copied to these bit positions.

When receiving, the received value is compared with the local counter, and if they differ, a correction is made.



Algorithm for Uniform Polling of Virtual Channels

Virtual channels must be basically synchronous. To obtain such channels, TDM (time division) mechanism is used, but this principle has a problem: the virtual channels are obtained either at the same rate (E1-E3) or in the form of bursts with some sufficiently long transmission period (SDH).

To comply with the transmission uniformity principle in each virtual channel, it is necessary to achieve a deviation of no more than two transmission periods, and with a guaranteed time distribution allowing the use of FIFO buffers with a size of less than two symbols.

In the preceding section, counters and a mechanism for maintaining its consistency with the transmitted sequence of symbols were created.

Counters have a certain period (for a specific case, we will assume that this is the number of symbols per second). A counter takes the value from zero to a number of symbols per second minus one, representing the equivalent of TDM with a time slot of one symbol per second.

Algorithm for Uniform Polling of Virtual Channels

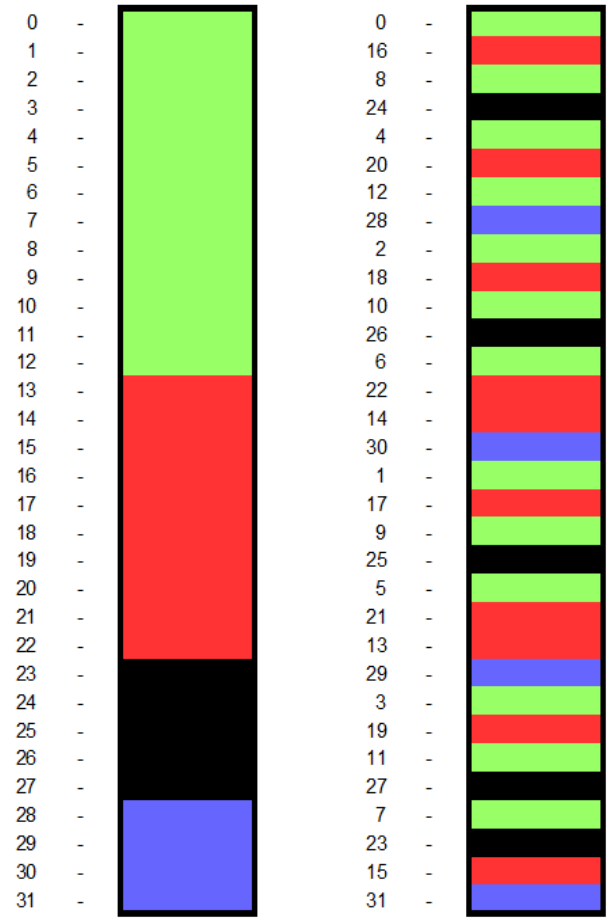
Each virtual channel occupies a certain number of the adjacent combinations (cells) in this space.

The number of combinations is proportional to the required synchronous component of a virtual channel rate (it is similar to the memory allocation in an ordinary computer).

The counter value, at the time a symbol is received, can be used for addressing a specific FIFO. Each buffer responds to a certain range of values specified when creating a channel.

This method has a problem: Data will be transmitted in bursts with a period of one second, which is unacceptable. An algorithm is needed to convert the value in a counter into FIFO number with a guaranteed uniform time distribution for any option of dividing the source physical channel into components.

A simple algorithm was found: it is enough to change the high-order bits of a counter for the low-order ones (mirror image), and there will be uniform access to any given continuous section of the values of an original counter with a period equal to the ratio of a symbol rate of a physical channel to the width of this section (it needs to be proven mathematically over the entire range of rates and partitioning options).



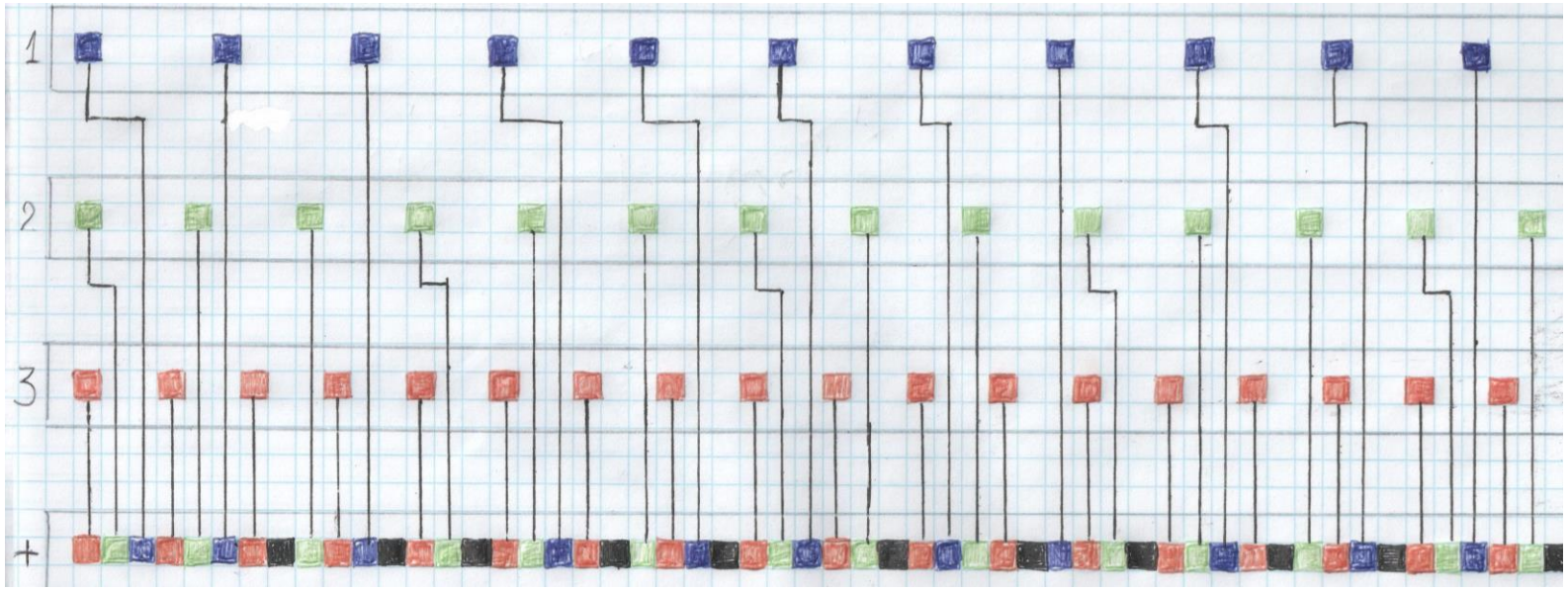
This method is completely insensitive to the channel creation errors when a request to create a channel did not reach a receiver (transmission, decoding error). In this case, the data will occupy time slots not related to any FIFO, and a controller should simply ignore it.

Algorithm of the Virtual Channel Frequency Method

You can allocate a bandwidth using another method. To do this, you need to sort out a list of the requests for creating the virtual channels in the ascending order of a desired rate.

Add data to the channel after the data transmission period of each virtual channel has expired (the principle is illustrated in the picture).

This algorithm is less efficient than the first one and requires special measures to protect against loss of requests for creating a channel.

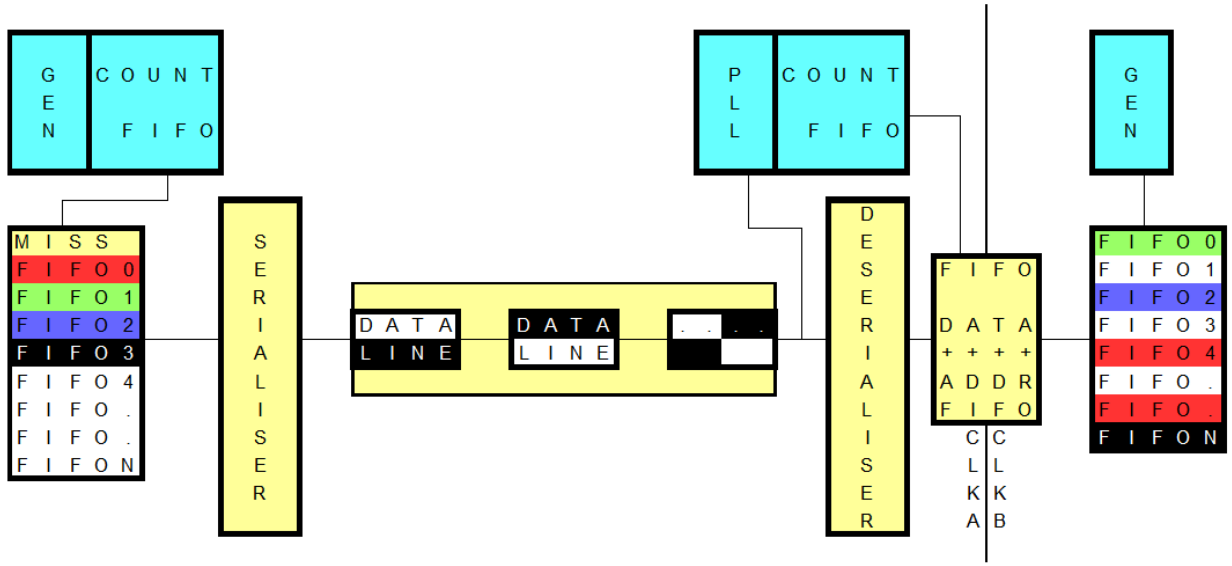


Creating, Deleting a Channel

The symbols being transmitted are divided into large groups, among which are: User Data (synchronous traffic) and Switch Control Data (asynchronous traffic).

Switch control data is transmitted using the bandwidth of a physical channel not used by a user, and it is possible to use both continuously unused part of a physical channel and a local one (at this particular moment).

The switch control data channel is asynchronous or an “asynchronous component source.” If, at the time of reading user data from FIFO, it turns out that there is no data, “no data” symbol is inserted. This symbol does not have to be sent to a channel; it can be replaced with the switch control service traffic.



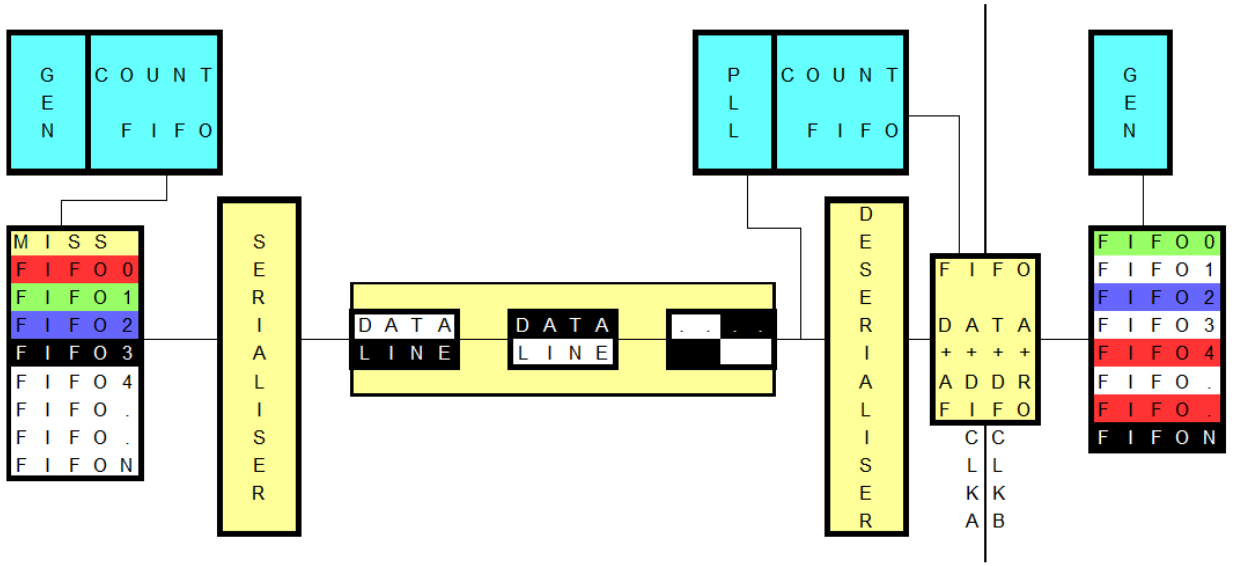
Creating, Deleting a Channel

On the receiving side, perform a reverse replacement; all service symbols are replaced with “no data” before writing to the receiver FIFO. This approach allows you to increase the efficiency factor of using a physical channel up to 98% (the remaining 2% is used to indicate the data type).

We get one fast channel with switch control data; therein you can transmit requests for creating the virtual channels and asynchronous component of user traffic.

After transmitting a request, send a service symbol “configuration change”, and further there is data in which a new channel is already present.

It is better to delete a channel by adding a user service symbol “delete channel”; after its transmitting to a physical channel, FIFO should be released, a buffer will stop responding to its address range (“no data” symbol will be transmitted).



Routing, Branching, Reservation of Channels

Routing can be of any kind (absolute or relative); relative one was chosen as base because it is easy to implement.

The receiver address is a sequence of output ports of switches through which the created virtual stream must be routed in order to reach a receiver.

If exactly one port is assigned to each switch, you will get a point-to-point communication channel.

If it is possible to redirect a stream simultaneously into two output physical channels, we get a tree-type structure (one transmitter - many receivers). In the previous slide you can see that one red virtual channel (the transmitting side) has turned into two channels on the receiving side. This effect is achieved by a fairly simple method: More than one FIFO buffer responds to one address range (to writing).

Writing occurs into several buffers, which are subsequently read for transmission to various virtual channels (including various physical channels).

The ability to control a specific route (relative addressing) helps ensure that multiple channels of data being transmitted will never be transmitted in one physical channel. In combination with the possibility of branching a virtual channel, we obtain the technology for multiple duplication of a communication channel with a proportional increase in the reliability of the virtual channel operation.

Creating a Virtual Channel Hierarchy

If, at the time of allocating time slots, an adjacent area for writing is allocated for several virtual channels, and for reading - the same area to be read as a single channel, we get a virtual channel containing several source and probably single channels.

Further the resulting aggregate channel can be transported using only one FIFO channel, but not several (according to the number of source channels).

After transmission, we perform the reverse procedure and once again get many source channels. The principles of working with an aggregate channel are not very different from the main one (physical channel).

Asynchronous Data Transmission

When creating a virtual data transmission channel, the asynchronous component of rate may also be in demand. For example, there is a process that sends data at previously unknown moments in time; only a medium rate of the transmitted data for sufficiently long period of time is partially known.

For maximum efficiency, it is required to deliver this data as quick as possible (with the specified latency). If there are only synchronous (at a constant rate) data transmission channels, you will have to allocate a bandwidth, proportional to the requested latency, to each such process and, in this case, an allocated channel will not be fully loaded.

To understand what factors the asynchronous bandwidth depends on:

- The source channel is occupied by the synchronous channels, there remains an unoccupied area and skips in the provision of synchronous data. It is almost impossible to predict the availability of these resources in advance, and this is a weakness of such asynchronous networks as packet switching.
- On the other hand, for using instant skips (pauses in the data transmission), a buffer is needed, in which the data will be located, in an amount in excess of what is required for the synchronous transmission.
- To create a virtual channel with synchronous and asynchronous components, it is necessary to occupy exactly as much synchronous bandwidth as it is required on average over a long time interval.

Asynchronous Data Transmission

- To provide an asynchronous component, increase the size of FIFO buffer in proportion to the ratio of the synchronous and asynchronous rate components.
- At a time when there is no data to transmit (a buffer is empty or time slots are not occupied), you can transmit data from the channel buffers with a non-zero asynchronous rate value, using a service part of the symbol set.
- There are two behavior options for the transmitting side. The first is to transmit at one time a number of symbols equal to the size of the allocated FIFO and wait for notification of receiving; a fully asynchronous mode working well over short distances (data center, supercomputer).
- For long distances (more than 100 km), it is more optimal to try to transmit data without rate limitations, but this will cause an overflow of FIFO buffers of the intermediate switches. With this transmission mode, FIFO, where an overflow occurs, must replace the last symbol in a buffer and a symbol that is trying to be written to the “Data Overflow” service symbol with a counter for the number of such data.
- On the receiving side, the adopted sequence is analyzed, and a request is made to retransmit the lost data.
- The transmitter, upon receiving such a request, will reduce the transmission rate and retransmit the requested symbols (a similar mechanism is used in TCP IP).

Interface with a Computer System

What might a network interface look like with different computing device options?

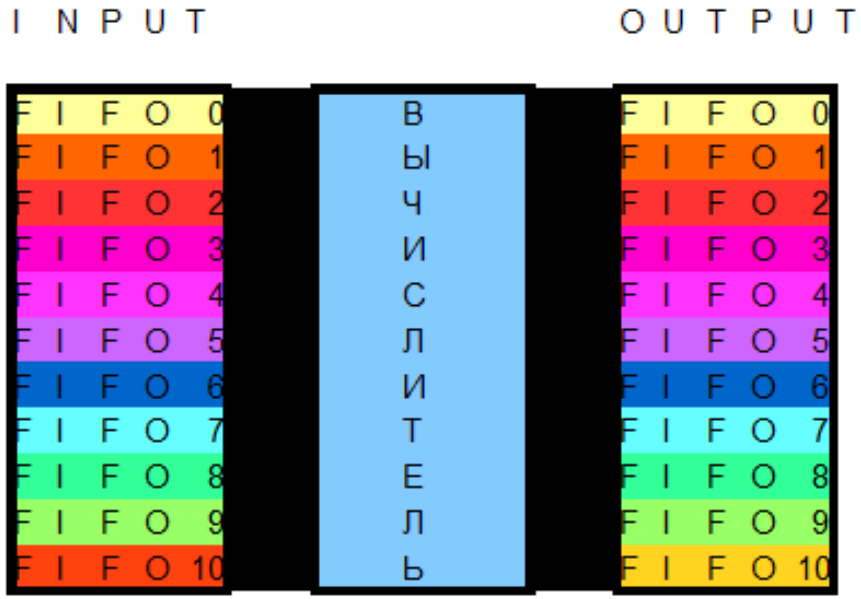
Where should a network-computer interface be located? It's all simple, the most convenient part is a switch. The switch is a large number of identical FIFOs, which are accessed through the physical channel modules.

The most effective way is to "imitate" another physical channel.

The simplest option is to disconnect the inputs or outputs of individual FIFOs from an interface to the physical channels and write (read) data using the hardware of a specific computer.

For ASIC, FPGA and something of the kind - everything is very simple; FIFOs are in fact registers with synchronous control that are very easy to interface with similar types of devices.

For computers with an interface based on the memory type (address, data), it is more optimal to "imitate", select a certain area in the address space, assign its own address to each FIFO, and, if it's possible, implement specific access control commands.



Besides, you can add an interrupt service that calls a handler (program) when data appears in the required FIFOs or when there is a threat of an overflow, etc.

Memory Disaggregation (DSM)

What the memory disaggregation technology might look roughly like? (there will be only a processor with cache memory and data transmission network chiplets on the motherboard).

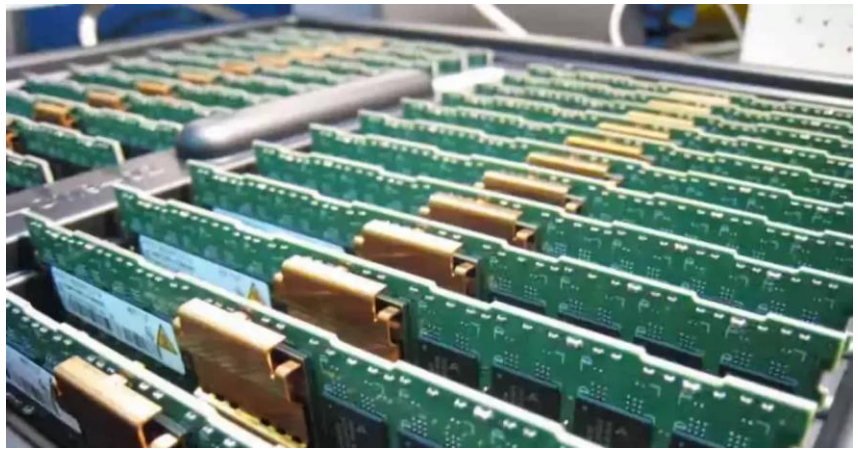
A memory has an access time of 50-70 ns; it is required to ensure similar delays in order “not to feel” the difference between local and remote memory.

We believe that a memory server carries out caching (approximately like a fourth-level memory cache), and in 99% of cases the access goes specifically to a memory cache.

For convenience, we assume that a processor consists of individual modules located in a three-dimensional cube with connections between the nearest neighbors (100 mm + 60 mm chip size).

There is a memory server at the center of this cube. Switching time is 1 ns (I think it’s achievable with a switch clock frequency of 5 GHz). We get an approximate time for transmitting a message to a neighboring module at the level of 2 ns (1 ns switching - 1 ns communication cable). Data reading from a memory cache - 10ns.

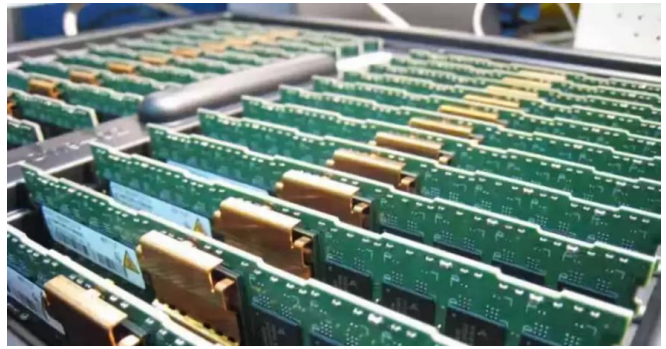
Simple calculations show the distance at which there is no difference between local and external memory, it equals about 5 meters. Moreover, the cube side is 1.6 meters, which is comparable in volume to a standard rack for telecommunications or computing equipment.



Considering the maximum connectivity with neighboring elements and reading by large pages, the memory access overhead costs will be less than local memory access latencies.

Memory Disaggregation (DSM)

What an approximate memory access scenario might look like:



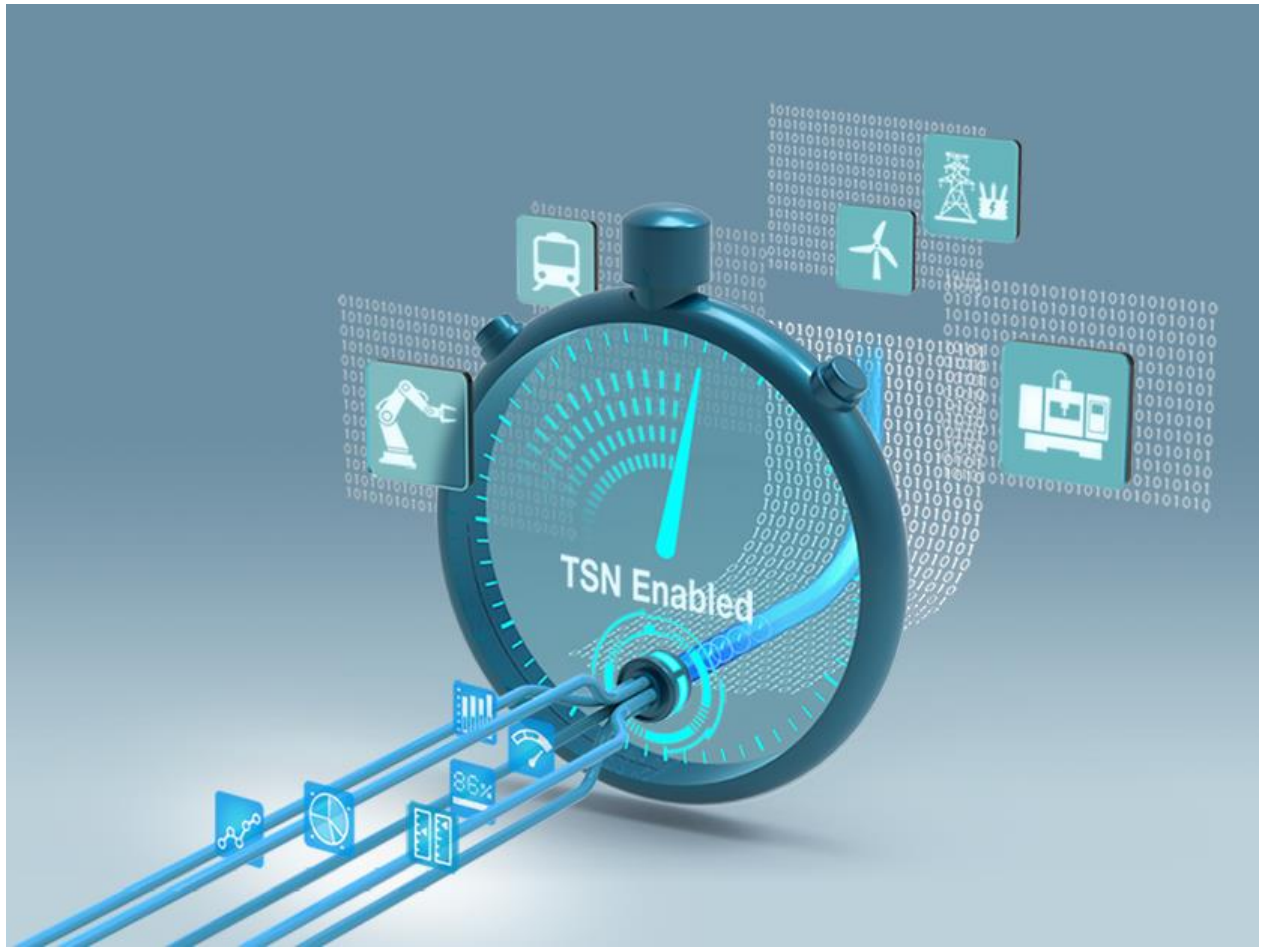
- There is a remote memory server, a processor creates a virtual channel with a large asynchronous rate component.
- A server responds by connecting the processing element to a single synchronous channel (at a high synchronous rate).
- When the work starts, a request is sent to fill a local cache. The page transmission occurs (for example, 4K), and it is simultaneously seen by everyone else who has “subscribed” to this memory area.
- If reading is required, we read from a local memory. If writing is required, a request to modify the memory cell is sent to DSM server. If there is no need for further reading of the modified cell, the computing process continues. Otherwise, a processor stops and waits for the cell content update notification via a high-rate synchronous channel.
- This distributed DSM memory option ensures that every element of the system sees the same sequence of changes in the contents of a shared memory.

Telecommunication Network for Real-Time Systems

Due to the guaranteed rate, pre-calculated and constant maximum latency, the ability of copying data, route control, all the requirements of real-time systems are met.

One could say that the proposed architecture of SSH network is generally the first and so far, the only implementation of all the requirements of a real-time network.

All others are either limited functional (TDM) or not real-time networks at all (any packet networks).



THANKS FOR YOUR ATTENTION, ANY QUESTIONS?

For more information you can contact:
Balyberdin Andrey Leonidovich **Rutel.Nsk@Gmail.com**