Avoid LAN Switches
IP Routers provide a better alternative for a Real-Time Communication System

E. Simon, E. Gressier-Soudan, J. Berthelin
CNAM-CEDRIC
292 rue St Martin
75 141 Paris Cedex 03, France
simon@host.fr, gressier@cnam.fr, jb@cnam.fr

Abstract

This paper describes how routers can be used for factory communications. This statement is based on QoS management features of the IP protocol, especially schedulers, shapers, markers, filters that can be used and the ability of IP datagrams to carry priority in the ToS field for Ipv4 or the DS field for Ipv6. This last property allows priority inheritance of real-time threads across a factory network. As a conclusion, switches could be used for devices connection, whilst routers could be dedicated to real-time priorities management. We describes in this paper our experiments on the use of iproute2 and tc in Linux to improve our knowledge of QoS management in routers.

1. Introduction

A few years ago we entered the Industrial Ethernet era for factory communication systems. Industrial Ethernet is a generic expression that gathers full-duplex switched Ethernet with the protocols TCP, IP, and sometimes UDP. Many papers in conferences related to Fieldbus Systems or Factory Communication Systems deals with the constraints of real-time communications over Ethernet (for example: ETFA'2001, FET'2001, WFCS'2002 are conferences where we went and where we encountered contributions on this topic). Also, Fieldbus Systems suppliers are moving toward Industrial Ethernet. Most of them generally sell a port or an adaptation of their fieldbus protocol on top of Industrial Ethernet in their product line. MODBUS [5] is may be one of the first that did a such work.

Does Industrial Ethernet suit the requirements of real-time communications? The answer, in the scientist community, seems to be definitively no. The problem is not only Ethernet despite the full-duplex switched mode extended with the 3 bits of priority from the 802.1p standard, but also TCP itself. It is well known that reliability has a cost, and that a standard TCP implementation with flow control and congestion control mechanisms introduces latency and jitter. UDP with RTP/RTCP [7] and a TCP friendly approach [8] could have been a very interesting alternate solution.

The aim of this paper is to advocate that Industrial Ethernet is not the wrong solution because of Ethernet or TCP, but mainly because it relies on switches. We deliberately state that switches are not well suited to support real-time communications. Our belief is that routers are going to play a key role in the future of factory communication systems. We do not completely eliminate switches in network design, they are just better than hubs. We believe that switches can't be anymore the key element for interconnection inside factory plants, living routers at the edge of the factory network to connect only the outside world. Routers have to play an important role because of their ability to support not only routing functions but also QoS mechanisms.

The paper is organized as follow. Section 2 deals with switches and routers. Section 3 recalls the QoS management features of QoS enabled routers. Section 4 advocates the use of DS and TOS fields in IP datagrams for end-to-end priority inheritance. Section 5 describes the iproute2/tc framework and our current prototyping. Section 6 concludes.

2. Message Forwarding: Switches vs Routers

Answers are given by Radia Perlman about bridges vs routers in [6]. In chapter 12 of her book, she states that bridges are chosen for their simplicity, and routers when routing through different paths is required. Switches are a new form of bridges, and what she said applies mostly to switches. But if the VLAN feature is considered in switches, factory communication systems become much more complex.

Generally, in a factory network, multiple paths are not needed except for fault tolerance. Routers perform route management better than switches with the spanning tree.

IP address management is also more flexible than Ethernet address management. The private address space

offers a large set of possibilities for naming devices. With the Classless Inter-Domain Routing approach, interesting features as subnetting have been introduced. Also Ipv6 will bring a larger and more flexible address space.

In addition, IP is able to handle multicast routing and forwarding, there is a rich set of protocols, the known ones are: DVRMP, PIM-DM for dense communities of multicast hosts, or PIM-SM that uses a shared tree with a Rendez-vous Point for sparse communities of multicast hosts. Rules are provided to map IP multicast addresses over Ethernet multicast addresses, the constraint is that IP multicast addresses shouldn't be too close because of collision risks on the resulting Ethernet multicast address (only the 23 lower bits of a IP multicast address are used to provide the Ethernet multicast address).

Therefore, the purpose of this paragraph is not to completely eliminate switch devices in factory communication systems. Switches are useful to interconnect devices but should not be the building bloc of a real-time factory communication system. We will see in next sections that the 3 bits for priority (8 levels, 2 are reserved) provided by the 802.1Q/p standard offers poor abilities to carry application priorities. Despite these disadvantages, we should underline the efforts of the industrial/scientific community to provide a Real-Time Industrial Ethernet.

3. QoS Management

Switches are very simple devices. As far as we know, there is no mechanism inside switches dedicated to traffic shaping, or scheduling. Such mechanisms exist inside routers of the DiffServ generation. Figure 1 depicts the architecture of a QoS enabled router.

The current functions encountered in a QoS enabled router are the following: classifier, scheduler, traffic conditioning in queues (metering, marking, shaping, dropping and queuing disciplines). In case of real-time communication management, a router should be as simple as possible: more functions mean more time spent to handle datagrams. All the previous functions are needed but not in broader way as for general purpose Internet networks. Classifier, and scheduler are mandatory, priority based mechanisms and FIFO queuing disciplines for each priority seems to be efficient to address real-time communications. Dropping is only used in case of congestion. Dimensioning and commissioning of a factory communication system would eliminate unpredictable streams and avoid dropping, the factory network is a closed world. It seems that marking is not useful, we can consider that datagrams carry their priority from the source to the destination despite the fact that, as far as we know, there is no API to tag a datagram with any priority. Metering can be implemented in one router for performance measurements, but it can also be simplified. Shaping is an interesting feature. It eliminates the jitter appeared in a message stream across a network. A message stream can become conformant to its initial profile (throughput, periodicity…). The cost of jitter elimination with traffic shaping is a bigger latency and the risk to fail in meeting application deadlines.

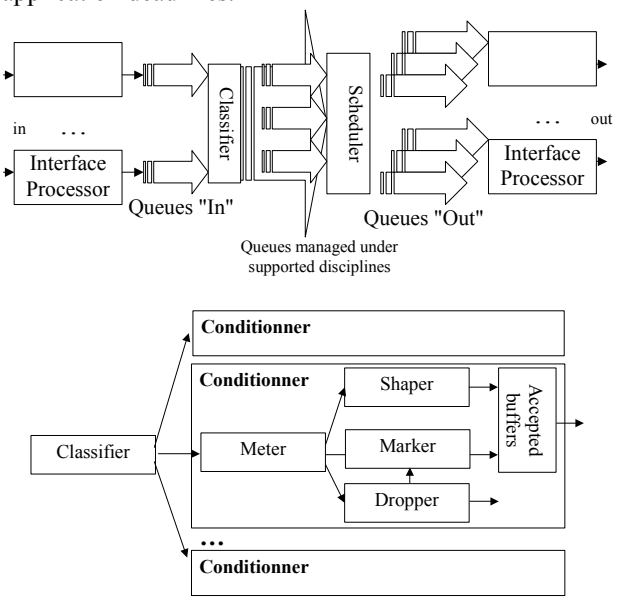

Figure 1. Mechanisms of a QoS enabled router

4. End-to-end Priority Inheritance across the network

4.1. DiffServ based Real-Time Factory Network Communications

Due to the DiffServ approach provided by the IETF for QoS routing, a field In the IP datagram, called the DS (Differentiated Service) field has been created. In the IPv4 datagram this field replaces the TOS byte (Type of Service) field that includes the four TOS bits, be carefull it is a little bit confusing. The DS field is 8 bits long. It contains two unused bits, and the DSCP (Differentiated Service Code Point) field. The different values for DSCP are defined in RFC2474. Different classes are defined: EF (Expedited Forwarding) class for the best quality of service, different AF (Assured Forwarding) classes, BE (the old world wide known Best Effort) class. IANA lets 32 values to distinguish classes, in a first approach, it is better than the 8 values (less 2 reserved for LAN management) provided by the 802.1p LAN priority management proposal.

This framework allows to route real-time communication by class. The key issue is then to group the different application traffic of the same profile and to allocate a class. It is easy to guess that the EF class will receive the more stringent traffic.

4.2. End-to-end priority inheritance

A previous distributed real-time platform project was able to deliver very good performances [4]. It was based on ATM technology and specific features developed inside the core of the real-time kernel ChorusOS. Despite its real-time properties, and the efficiency of its implementation, its major drawback was its specificity, and couldn't be used in real life projects.

One of its key features was the ability to carry priorities in ATM cells across the network. This opportunity came from one of the AAL3/4 field, 6 bits were re-used to carry 64 levels of priority. It was also possible because the company that led the project was also the ATM board designer.

With the IP protocol, and the DiffServ approach, we are now able to do an equivalent work more easily. The only constraint is: any priority 000000xx (four kernel priorities in fact) should map the best effort class of service and can be handled the same way. There are two alternatives: Backward compatibilities with the admitted tags of the DiffServ approach adopted by the IETF (Expedited Forwarding and assured Forwarding classes) could be preserved. In this case 32 levels of priority are available. A more aggressive scheme can be adopted. Inside the routing domain of a factory plant nothing forbids the free use of the full 8 bits of the DS field to contain a priority. 8 bits lead to 256 levels of priority, exactly the same as the number of priority levels supported by most of the real-time kernels.

We believe that the overall DS field of the IP datagram (either Ipv4 or Ipv6) offers promising possibilities. The difficulty is to provide a standard framework at low cost that allows the end-to-end priority inheritance feature. We believe that the iproute2 and traffic control tools under Linux are convenient to achieve this goal.

5. Prototyping QoS management features of the Linux kernel

The iproute2/tc framework [3] allows to prototype the previous mechanisms inside Linux routers. iproute2 is a kernel package that unifies network related tools (previously arp, ifconfig, route, and new tools as tunnels). The Linux traffic control (tc) tool allows the configuration of the IP stack through iproute2 features to enable QoS management inside the kernel. It especially deals with queuing disciplines. These facilities promote Linux as a cheap, easy to use basis to build the routing architecture of a factory network.

5.1. The iproute2/tc Framework

This framework allows the management of different queuing disciplines, classes and filters. Some of the features implemented are: CBQ (Class Based Queuing), HTB (Hierarchical Token Bucket), SFQ (Stochastic Fairness Queuing), TBF (Token Bucket Filter)… Figure 2 gives an example of queuing management provided by iproute2/tc. And figure 3 gives the corresponding configuration using tc.

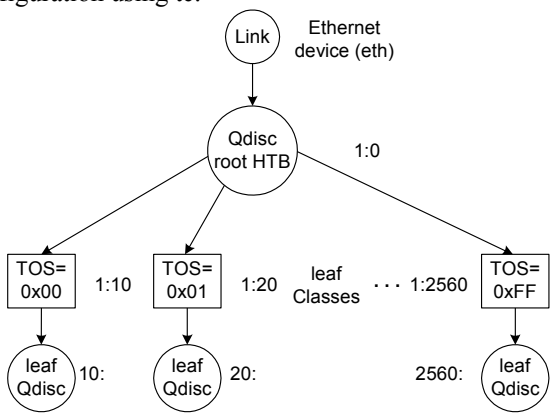

Figure 2. Queuing management with iproute2/tc

```
# root queue initialization
tc qdisc add dev eth1 root handle 1:0 htb
# Leaf classes
CLASS=class add dev eth1 parent 1:0 classid
# Class TOS=0x00
tc $CLASS 1:10 htb rate 100kbit burst 15k prio 0
# Class TOS=0x01
tc $CLASS 1:20 htb rate 100kbit burst 15k prio 1
# Class TOS=0x02
tc $CLASS 1:30 htb rate 100kbit burst 15k prio 2
…
# Class TOS=0xFF
tc $CLASS 1:2560 htb rate 100Kbit burst 15k prio 255
# leaf Qdisc configuration
tc qdisc add dev eth1 parent 1:10 handle 10: pfifo
tc qdisc add dev eth1 parent 1:20 handle 20: pfifo
tc qdisc add dev eth1 parent 1:30 handle 30: pfifo
…
tc qdisc add dev eth1 parent 1:2560 handle 2560: pfifo
# U32 filters configuration
FILTER="filter add dev eth1 protocol ip parent 1:0"
tc $FILTER prio 0 u32 match ip tos 0x00 0xff  flowid 1:10
tc $FILTER prio 1 u32 match ip tos 0x01 0xff  flowid 1:20
tc $FILTER prio 2 u32 match ip tos 0x02 0xff  flowid 1:30
   …
tc $FILTER prio 255 u32 match ip tos 0xff 0xff  flowid 1:2560
```

Figure 3. Queues configuration example using tc

The command file configures a Linux kernel linked by 100Mb/s Ethernet LANs. The HTB scheduler is used, a class based strategy more efficient than CBQ [2]. 256 queues are defined and handled on a priority basis. Each queue supports a 100k/s throughput. The u32 filter is used to classify the forwarded datagrams in the right queue. The "pfifo" parameter defines a FIFO discipline, statistics are provided on a per datagram basis. iproute2 is very flexible despite its awful syntax. The reading and

understanding is really a hard stuff for the beginner. Figure 2 is a good abstract of the organization of queues.

5.2 Accessing the TOS field

To deal with priority within the 8 bits of the TOS field it is necessary to access it. This is important because the value persists across the network. There is two ways. The first one compatible with the DiffServ approach is based on the use of the DSMARK queue discipline of tc [1]. The second one is a more open solution, it relies on the iptable/netfilter tool. We skip the details of this tool, the main idea is to mark the IP datagram within the Linux kernel with the required priority on any host.

Things are not so easy. A first glance at the network code of Linux exhibits new difficulties. The allowed values for the DS field (or the TOS field) correspond to the standard use specified in IETF's RFCs. As a consequence, it is not possible to mark this field with priorities ranging from 0 to 255 as we would like without modifying the Linux kernel itself.

Past experiments have made a deep use of iproute2/tc with different queuing disciplines and schedulers. The next step is to modify the network protocols. We'd like to alleviate some controls on the DS/TOS field to be able to use our new feature of 256 priorities and to answer real-time factory communication system needs.

6. Conclusion

Radia Perlman was right when she said that switches are simpler to handle than routers.

We believe routers are an important piece of the puzzle for factory communication systems. Routers are QoS enabled while switches are not. Also IP, with the DS field is able to support a rich set of priorities for real-time thread-to-thread communications with priority inheritance. These are key issues to address real-time requirements in factory networks.

Finally, routers are the only devices able to support the required mechanisms to ensure QoS guarantees in applications. Therefore, routers are the key element to design factory communication systems and to address priority inheritance based real-time thread-to-thread remote communications.

References

[1] W. Almesberger, JH. Salim, A. Kuznetsov. Differentiated Services on Linux, June 25, 1999. IETF draft-almesberger-wajhak-diffserv-linux01.txt Available via http://icawww1.epfl.ch/linux-diffServ/

[2] M. Devera. HTB Home page. http://luxik.cdi.cz/~devik/qos/htb/. June 2003.

[3] "Linux Advanced Routing & Traffic Control HOWTO". http://lartc.org/lartc.pdf. April 2003.

[4] C. Lizzi, L. Bacon, E. Becquet, E. Gressier-Soudan. "Prototyping QoS based Architecture for Power Plant Control Applications". IEEE Workshop on Factory Communication Systems (WFCS'2000), September 2000, Portugal.

[5] Modbus/TCP Homepage. Automation Business. http://www.modicon.com/openmbus/. May 2001.

[6] R. Perlman. Interconnections: Bridge and Routers. Addison-Wesley. 1992.

[7] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," Request for Comments (Proposed Standard) 1889, Internet Engineering Task Force, January 1996.

[8] D. Sisalem, A. Wolisz. "LDA+: A TCP-friendly adaptation scheme for multimedia communication". IEEE International Conference on Multimedia and Expo III. P 1619-1622, 2000.