

Advanced traffic management aids converged IMS applications

IP Multimedia Subsystem (IMS) describes the network infrastructure that supports emerging Internet Protocol (IP) multimedia and telephony applications. IMS is defined by the 3rd Generation Partnership Project (3GPP) standards for wireless networks and is also being applied to wireline networks.

As applications based on IMS are becoming widely available to consumers, the equipment deployed in the traditional best-effort Internet will need to be upgraded to support the increased bandwidth and stringent Quality of Service (QoS) requirements.

To support IMS, new wireless cell phones and multimedia platforms will integrate combinations of technologies, including:

- Bluetooth
- Wireless USB
- Ultra Wideband
- 2.4 GHz and 5 GHz wireless LANs
- GPS location
- High-Speed Downlink Packet Access (HSDPA)
- True mobile video broadcasting technologies

For wireline terminals, high-definition video, DSL, and Passive Optical Network (PON) enable and drive high-bandwidth IMS applications.

Nonblocking architecture essential

With real-time video becoming more widely deployed, real-time traffic can account for significant network bandwidth, which can lead to starvation of data traffic. As such, a combination of strict priority and minimum bandwidth guarantees will become crucial and the system must be nonblocking to support deterministic QoS.

To realize the required QoS of a mixture of traffic with various Class of Service (CoS), advanced traffic management architectures are often used. The challenge is that QoS guarantees must be preserved in the presence of network congestion, component

board failure, various CoS requirements, and multicast traffic.

In order to meet these goals, advanced traffic management often makes use of the following techniques:

- Prioritization and isolation of traffic based on CoS using queuing architectures
- Congestion management using flow control mechanisms that act on queues
- Allocation and management of resources (bandwidth, buffers) to optimize system performance

Network element systems that implement advanced traffic management often support multiple ports, which are used for aggregation or interconnection of various networks together in the Internet. A popular standard interface used inside such systems is RapidIO. Defined by the RapidIO Trade Association, RapidIO is an industry standard fabric interconnect

supplying the advanced traffic management that converged IMS applications need.

RapidIO is designed to be compatible with popular integrated Network Processing Units (NPUs), communications processors, host processors, and networking DSPs. RapidIO enables nonblocking switching and guaranteed QoS. Table 1 shows the system requirements for achieving guaranteed QoS, and how RapidIO addresses these requirements.

RapidIO features that enable low latency include:

- Low overhead protocol resulting in high fabric utilization
- Hardware-based data encapsulation with a small Maximum Transmission Unit (MTU) size for reduced latency variability
- Hierarchy of flow control features for congestion avoidance and congestion management

Requirements	How RapidIO addresses requirements
Low system-level latency	Lightweight protocol stack implemented in hardware
High link bandwidth to support broadband applications	1 and 4 lane SerDes-based serial links support 2.5 and 10 Gbps data rates.
Next generation physical layer	Next generation physical layer SerDes with support of up to 80 Gbps data rate per port
Minimum bandwidth guarantee	Virtual Channel (VC) architecture consists of scheduling and flow control mechanisms, which result in isolation of traffic from diverse classes of service.
Nonblocking system	System-level Virtual-output-Queuing (VoQ) and flow control enable real-time QoS support
Fine granularity traffic management	Streams with a few Kbps bandwidth granularity are managed individually.
End-to-end flow control and resource management	End-to-end logical layer flow control between processors
Multicast	Transport layer supports up to 64K connections that can be used as unicast connections or to designate multicast groups. Switches are capable of multicast.
Interworking with other packet interfaces with QoS support	Data streaming encapsulation and physical layer (segmentation and reassembly)
High availability, redundancy, hot swap, and failover	Fault detection and dual star configuration
In-order packet delivery	Supported by physical layer
Lossless transmission at the link and end-to-end levels	Supported by physical and logical layers

Table 1

RapidIO offers hierarchical flow control architecture at the link and logical layers. The link layer's transmitter based flow control makes pipelining effective. Credit based Virtual Channel (VC) and Virtual-output-Queuing (VoQ) flow control are also supported. The logical layer (processor to processor) supports Xon/Xoff of specific flows, bidirectional arbitration for flows, and end-to-end flow management of classes, streams, and ports.

The associated flow control mechanisms of the VC and VoQ architecture operate at the link layer. What's more, they can provide dramatic performance benefits at the system level, while adding minimal overhead. VC architecture allows for minimum bandwidth reservation for each class of service. VoQ architecture allows for system-level head-of-line blocking avoidance to guarantee deterministic latencies for real-time traffic.

VoQ and VC architectures

Consider a one-lane road with cars arriving at an intersection. If the first car in line should turn left, and is blocked from doing so (due to heavy traffic in the left road), then all of the cars behind the first car must wait, even though the other roads (straight and right) are clear of traffic.

Translating this example to a communication system, the cars represent packets, the intersection represents a hub switch, and the backed-up line represents a First-in-First-out (FIFO) queue. Real-time packets that are blocked in the FIFO would suffer latencies, which are non-deterministic. The packets would likely fail to achieve their required levels of QoS. Examples of standard protocols, which suffer from system-level blocking, are Ethernet and PCI Express.

On the other hand, consider a road with multiple lanes. Cars arriving at the intersection go to the lane corresponding to the turn they plan to make. If the car turning left is unable to do so because of congestion, the cars in the other lanes are still able to go because they have another path to their destination using the other lanes. Implementing the RapidIO VoQ architecture enables a nonblocking system.

To understand the advantages of the Virtual Channel architecture consider a road with a continuous stream of cars passing through an intersection that have the right-of-way because they are on the

main road. If the car on the side street arrives at the intersection and does not have the right-of-way (for instance, if the vehicle encounters a stop sign), it must wait for a gap in the main street traffic in order to turn left. This wait could be indefinite, which can lead to starvation.

In a communication system, real-time packets, are usually given the highest priority in order to guarantee the lowest latency. With the resulting strict-priority scheduling, a data packet, must wait until there are no other higher priority packets (such as real-time packets) destined for the same output before it can be transmitted. Thus, data packets can experience starvation, receiving zero bandwidth for a potentially long time. The Ethernet protocol does not specify the scheduling for its priority queues. PCI Express provides the option of minimum bandwidth for its VC queues. Implementations of only strict priority for these standard interfaces will suffer system-level blocking.

Combination approach

On the other hand, consider that data packets will be scheduled to be transmitted at particular intervals. Thus, data traffic will not be starved and can be guaranteed a minimum amount of bandwidth specified by the QoS requirements of the particular CoS to which the data packets belong. The RapidIO VC architecture allows for the system to guarantee minimum bandwidths, while also preserving the low latencies required for real-time packets. To do this, often a combination of strict and minimum bandwidth scheduling will be employed. VC and VoQ architectures can be implemented in a communication system to improve system performance.

A communication system, typically consists of several line cards. The line cards communicate to each other as peers through a shared hub switch. Each line card may have a single processor or multiple processors used as a farm. The processors are usually connected to a bridge or a local switch on the line card, which is in turn connected to the hub switch over a RapidIO backplane. The interface between the processor and the bridge and between the processors and the local switch can be RapidIO as well in order to reduce system cost and improve performance.

In order to support VoQ, the local switch allocates a separate queue per hub switch output port. In this simple example, if the

hub switch's output port 1 is congested, the hub switch conveys VoQ flow control information to the local switch, which should halt the traffic in the queues corresponding to hub output port 1. The local switch's multiple queues, one per hub switch output port, correspond to the multiple lanes at an intersection for cars going left, straight, and right. The link between the local switch and the hub switch represents the intersection. If any one of the hub switch's output ports is congested, the packets in the other queues of the local switch are not affected and can still proceed as usual.

Similarly, to support the virtual channel architecture in RapidIO, a separate set of queues is allocated for each VC. While RapidIO provides support for up to nine VCs. In this simple example, there are two VCs. In the local switch, each VC hierarchically supports a group of VoQs for the hub switch's output ports.

The hub switch, which can maintain the same queuing architecture as the local switch, should schedule packets for each VC according to its minimum required bandwidth. VCs that correspond to data applications are usually the most in need of minimum bandwidth guarantees. Real-time traffic scheduled according to strict priority could grab much of the system bandwidth and potentially starve data packets indefinitely. However, the scheduler must also make sure that interactive and streaming packets are transmitted with the required levels of low latency. Erlang has implemented schedulers that support the requirements noted earlier as a combination of strict priority and minimum bandwidth scheduling.

If the hub switch experiences excessive congestion for a particular VC, flow controlling that VC lets traffic belonging to the other VCs traverse the system without violating their QoS requirements. Control takes place by having the hub switch convey VC flow control information to the local switch for a particular VC. The local switch's scheduler will then react to the flow control by limiting the number of packets transmitted for that VC. This can only be done if the local switch has queued packets for each VC separately. If the hub switch experiences congestion for VC1, it sends flow control information to the local switch, which temporarily stops transmission of all packets belonging to VC1 queues.

Technology Update

To model IMS systems with a mixture of voice, video, and data traffic, it is important to simulate bursty traffic. Data transfers and variable bit-rate video traffic tend to be bursty in nature. The simulation assumes an average burst length of 20 packets being transferred between line cards. If each RapidIO packet size is 256 bytes for the payload, this would correspond to $20 \times 256 = 5,120$ bytes being transferred, which could be the size of four video packets (each of 1,280 bytes), transmitted in a burst.

For the base case, consider a system architecture that does not support any flow control. The system implementation could be simpler and may be able to achieve the lowest latency at very low loads. For instance, at 1 percent load, it was assumed that this switch architecture has a fall through latency of 200 ns. However, with increased levels of congestion and the inability to control the conges-

tion, the latency is about 100 microseconds at 55 percent load and quickly goes to infinity thereafter.

A switch architecture that supports VC flow allows for improved performance, with about 100 microseconds latency at 65 percent load. This is an incremental performance improvement, however, the VC architecture allows for minimum bandwidth guarantees for traffic that is not real-time (data), while preserving the QoS for real-time traffic (voice and video). This can be achieved with a combination of strict, plus Deficit Round Robin (DDR) or Weighted Round Robin (WRR) scheduling, which Erlang has implemented in switch fabrics and traffic managers.

Finally, the switch architecture, which supports VC and VoQ flow control, starts at 800 ns latency at 1 percent load because the scheduling algorithm

requires more start-up time in the pipeline. However, the latency will be lower than the base case at loads greater than 25 percent and can reach 80 percent or higher loads. With additional scheduling and bandwidth management optimizations, Erlang has been able to achieve 95 percent and greater loads with less than 100 microseconds latency in high-performance switch fabrics.

Peter Yan is an active member of the RapidIO Trade Association and chief technology officer at Erlang Technology.

To learn more, contact Peter at:

Erlang Technology

345 Marshall Avenue, Suite 300
Saint Louis, MO, 63119

Tel: 314-276-5583

Fax: 314-336-5902

E-mail: petery@erlangtech.com

Website: www.erlangtech.com