NETSEDGE RESEARCH GROUP

Virtual Routing:
What's The Goal? And What's Beyond?

Peter Christy, NetsEdge Research Group, August 2001

Virtual routing is a software design method used to provide multiple "independent" routers that share a common processor, memory, and a set of interfaces. This in-turn supports an emerging class of internetworking device, a so-called "router aggregation system" at the network edge. There is some debate as to whether virtual routers are the best means to support customer demands for IP infrastructure outsourcing, or if a newer option based on 'multi-routers' is the way forward. This paper contrasts these two alternatives in light of hardware and software evolution.

Router Aggregation Systems

Router aggregation systems are designed to associate as set of interfaces with one of many routers with each router within the system being independently configured (including the protocols supported by each router). Rather than the resources at the edge being strictly partitioned (lines via interfaces attached to specific routers), they can be flexibly configured on demand. This greatly reduces the cost of configuring and maintaining these devices, as well as making much better use of the router resources (since an interface can be "moved" where it's needed).

Router aggregation systems are clearly an important idea due to significant cost of ownership benefits, partially from the better use of hardware resources, but largely because of the on-going savings in administration and operation. But what's the best way implement a router aggregation system given the various design tradeoffs?

This paper contrasts two approaches: one based on software virtualization ("virtual routers") and the other a multi-computer design ("multi-routers") based on a unique combination and association between hardware and software. Although these concepts are new to router aggregation systems, the basic design alternatives have been explored and contrasted in earlier applications.

Virtual routing today is accomplished primarily through software programming by virtualizing software processes in order to ensure maximum utilization of available resources. The good thing about programming is anything "can" be done. The bad thing about programming is almost every task is more difficult than it seems. The question is *not* whether virtual routing is a plausible design approach, but rather whether it is the logical design choice to make, given the inherent design requirements, such as the ability to concurrently support multiple autonomous systems and need to scale control plane processing.

The concept of creating virtual hardware resources has existed for quite some time, first appearing around 1965 in the form of the first "timesharing" systems -- a means of sharing very expensive mainframe CPU and memory. At that time the CPU and a megabyte of RAM *each* cost (literally) about $1 million, so there was a very high incentive to share. However, sharing hardware resources came with very clear tradeoffs relative to the performance and scalability of the individual programs. To get the most out of the hardware, you needed to use it directly.

Since that time, sharing resources, along with the associated tradeoffs, has continued. When DOS evolved to Windows, for example, the last programs to make the transition were games. This was because games made the most intense demands on the hardware resources and were the least amenable to multi-programming – the same multi-programming that made it possible to have a Word and Excel window open concurrently.

So how well does software virtualization work for routers? Is this an effective way of implementing a router aggregation system in order to save space and share resources among all the routers?

Router Basics

A typical router is a very valuable software application built largely on commodity computer hardware and packaged as a black-box, network appliance (rather than as a conventional computer system). Today's high-end routers are built on very specialized hardware, but low and mid range routers use standard computer components.

The first "routers" were software applications running on UNIX systems. Cisco, quickly pioneered the idea of routing appliances. Building a router as an appliance has the benefits of concealing the operating system (e.g. UNIX or Windows/NT) from the user and permitting the software to be optimized for the router (conventional operating systems are less efficient at tasks like protocol processing). In the end, a typical router boils down to a specific, but complex application, typically running on conventional computer hardware, with a specialized "real-time" operating system and specialized I/O devices (e.g., forwarding engines at the interface points).

Implementing a Router Aggregation System

The first router aggregation designs were software adaptations of existing router hardware because that was the easiest way to explore the opportunity – timesharing router systems[1]. This approach quickly became popularized as virtual routing. In some, relatively undemanding cases, this approach can work adequately, but there are many inherent problems:

[1] The term "virtual router" has been used earlier to describe applications that aren't really routers, such as in dial-up aggregation systems where the virtual router simply assigns the correct gateway and DNS assignments for each dial-up (or DSL/cable) user, but isn't capable of performing most of the functions we associate with a router. We will use the term "virtual router" here only for something that really performs router function.

- The total CPU power of a virtual routing aggregation system is approximately the same as the CPU power of a single router. In the mainframe days, more money could buy a faster processor. Today, the speed of a microprocessor is largely determined by semiconductor technology. The fastest microprocessor is more or less the same speed as the PC you would buy to run video games. Consequently, the CPU used in a virtual router aggregation system is unlikely to be much more powerful than that of a single router. The implication is if any of the router configurations in the aggregation system are CPU intensive, then the CPU power for all the virtual routers will be taxed. A virtual router configuration with eight virtual routers, for example will have one-eighth CPU capacity of eight independent routers. At most, each virtual router will have less power than a single router (making the optimistic assumption that all of the other virtual routers are very undemanding on CPU power and taking account only for the CPU power lost in the sharing). The conclusion is clear: virtual routers are not a good design choice for CPU demanding tasks such as calculating and recalculating large BGP (Border Gateway Protocol) routing tables or scaling I/O at the edge of the network.
- Design issues or implementation bugs that cause one router configuration to hog the CPU induces performance problems in other routers sharing the same CPU (see the following discussion on the general issue of software isolation and robustness).[2]
- The reliability of each virtual router is considerably worse than the reliability of an individual router. The real issue is software reliability, not the intrinsic reliability of the router hardware. A failure or reboot of any of the virtual routers within the system effectively disrupts the operation of the other virtual routers since they share a common system core. If each router reboots, on average, once a week, then eight virtual routers will each restart roughly once per day. In turn, virtual router systems are much less reliable than the independent routers they replace.

---

[2] A significant part of the discussion here concerns issues of software robustness and the value of separating each router processor as robustly as possible from all other routers, specifically using custom logic to enforce the separation at the hardware interface level. One might reasonably ask whether this is worth all the trouble. Software history suggests it is very much worth the trouble.

When the Java language was invented (as a language called Oak within the Sun Labs "Green" project) the primary goal was to achieve software isolation without using a "heavyweight" programming language. Java is carefully designed so that programs written in Java can not access program content or data elsewhere. In contrast, it's amazingly easy for a programmer writing in assembly language, or C or C++ to inadvertently cause these problems – mistakes with pointers and memory allocation being the biggest single category of such problems. Although it's taken quite some time for the completeness, efficiency and general maturity of Java to catch up with existing languages such as C++, in the end, these fundamental improvements in software robustness will drive its broad acceptance.

- Worse still, software errors in one virtual router may affect the operation of another virtual router. Software problems, such as memory leaks or pointer errors, have subtle and unclear indirect impacts. In conventional time-sharing systems (such as UNIX) one process is protected from another at considerable overhead cost by using heavyweight protection mechanisms. This comes at a considerable cost in context sharing and switching delays. If such an approach were used in a virtual router, more CPU cycles would be stolen and unavailable for the aggregate operation. Configuration errors or other malfunctions in one virtual router will impact performance and even the correct operation of other virtual routers.
- In a virtual router aggregation system, each of virtual routers must use the same version of the underlying operating system. With routers, a move to a new operating system revision typically occurs only when that change is known to solve a specific problem that has been encountered or in order to use a feature only supported in the newer release, since the changes can destabilize features that already work. In a virtual router configuration, the alternatives are either to move all the virtual routers up when any one router needs a new release or to require all routers to operate on relatively old and stable releases. Any system upgrade required for one virtual router will require a system disruption for all virtual routers (in effect, a reduction in the system availability for each virtual router). Virtual routers are also more constrained in terms of software revision level choices. This is a particular problem when customers using different virtual routers have very different network configurations and usage requirements.
- The memory of the aggregate router may be crowded. For instance, it takes several megabytes for a single router to store a single copy of the full BGP Internet routing table. These routers must also store the tables of their neighbors, many of which have hundreds of "next hop" neighbors. A router with 200 neighbors would then require several megabytes of memory for each neighbor along with enough processing power to compute the best topology for every possible route among all its neighbors. Consequently, routing can become a very memory intensive computer application as router software makes tradeoffs between execution speed and memory efficiency (memory is cheap, so use more), and because of the complexity of router databases (the complexity of networks themselves). The size of usable memory is limited by the memory addressing of the microprocessor as well as the memory interface support chips. If a single router can fully utilize the memory addressing capability of the hardware, then sharing that same memory among eight virtual routers will constrain the configuration of the other virtual routers. The use of virtual routers becomes problematic when any of the virtual routers has a particularly memory-intensive configuration.

Multi-Router Design Alternatives

In contrast to the software-multiplexing of a "virtual router" approach, an alternative design for router aggregation systems, known as a "multi-router" design, leverages multiple computer systems running concurrently. Rather than multiplexing multiple, logically independent routers on one central computer system (CPU and memory), multi-routers have multiple CPU/memory subsystems. Each router in the system has its own independent CPU, memory, I/O, and dedicated bandwidth to line cards.. This approach solves the problems described with virtual routing:

- Scalable CPU power so each router has its own, unshared CPU.
- CPU hogging of one router doesn't effect the operation of others.
- Software reliability is enhanced. Software problems on one router are isolated from other routers in the system. The mean time between disruptions of each router is essentially that of an independent router and is not diminished by the fact that multiple routers are running in the same box.
- Bug isolation: It is very difficult for a software error in one router to impact the functioning of another router
- Each router can run a different version of the base software.
- The use of memory for each router is unaffected by the needs of another.

Multi-Computer Systems

The architecture style used in multi-router systems more closely mirrors the mainstream server architecture that is now popular. Today's Internet technology applications require scaleable systems with multiple processors. The efficiency of these multi-computer systems can be significantly improved by not having multiple independent system boxes, each with its own power supply and I/O bus. Instead, new multi-computer designs are built around a single physical system consisting of multiple computer system "blades" (CPU and memory), and a common I/O system that connects all computers to common shared I/O (e.g. using InfiniBand system bussing). New multi-server systems take a mainstream system design approach and look very much like a emerging multi-router aggregation systems.

Hardware Design Problems Unique to a Multi-Router

Multi-routers require some hardware capabilities not needed by most multi-computer systems, such as rapid route table lookup. Whereas most I/O in a multi-computer is DMA (block transfer) oriented, the forwarding engines on the line interfaces need fast access to the route tables. Since each interface must be capable of accessing the route tables of any of the routers, some hardware mechanism is needed to make this fast lookup work without weakening the software robustness unnecessarily.

Summary

Router aggregation systems – hardware configurations capable of flexibly implementing multiple, independent routers – are an emerging and important form of network hardware designed to significantly lower operating cost, simplify network architectures and increase router efficiency at the edge of carrier networks.

Router aggregation systems can be implemented as a software adaptation of conventional routers, so called "virtual routers," or with multiple computer systems known as "multi routers." For all but the least demanding router applications, the multi-router approach offers significant benefits, as well as being consistent with the architectural evolution path of newer server systems.

NetsEdge Research Group
399 Main Street
Los Altos, CA 94022

John Katsaros
650-949-3256
john@netsedgeonline.com

Peter Christy
650-559-2103
pchristy@netsedgeonline.com

NetsEdge Research provides marketing and strategy research reports and consulting services, specializing in areas related to Internet infrastructure. NetsEdge offerings combine the formidable and unique experience and perspective of the two principals: John Katsaros and Peter Christy. At NetsEdge Research, John and Peter are continuing the work they began at the Internet Research Group (sold to Jupiter Research in 2000). At IRG, they produced definitive, early studies of the emerging markets in Internet Caching, Traffic Management, and Content Delivery and Distribution. Their clients for this work included most of the participants in the markets.