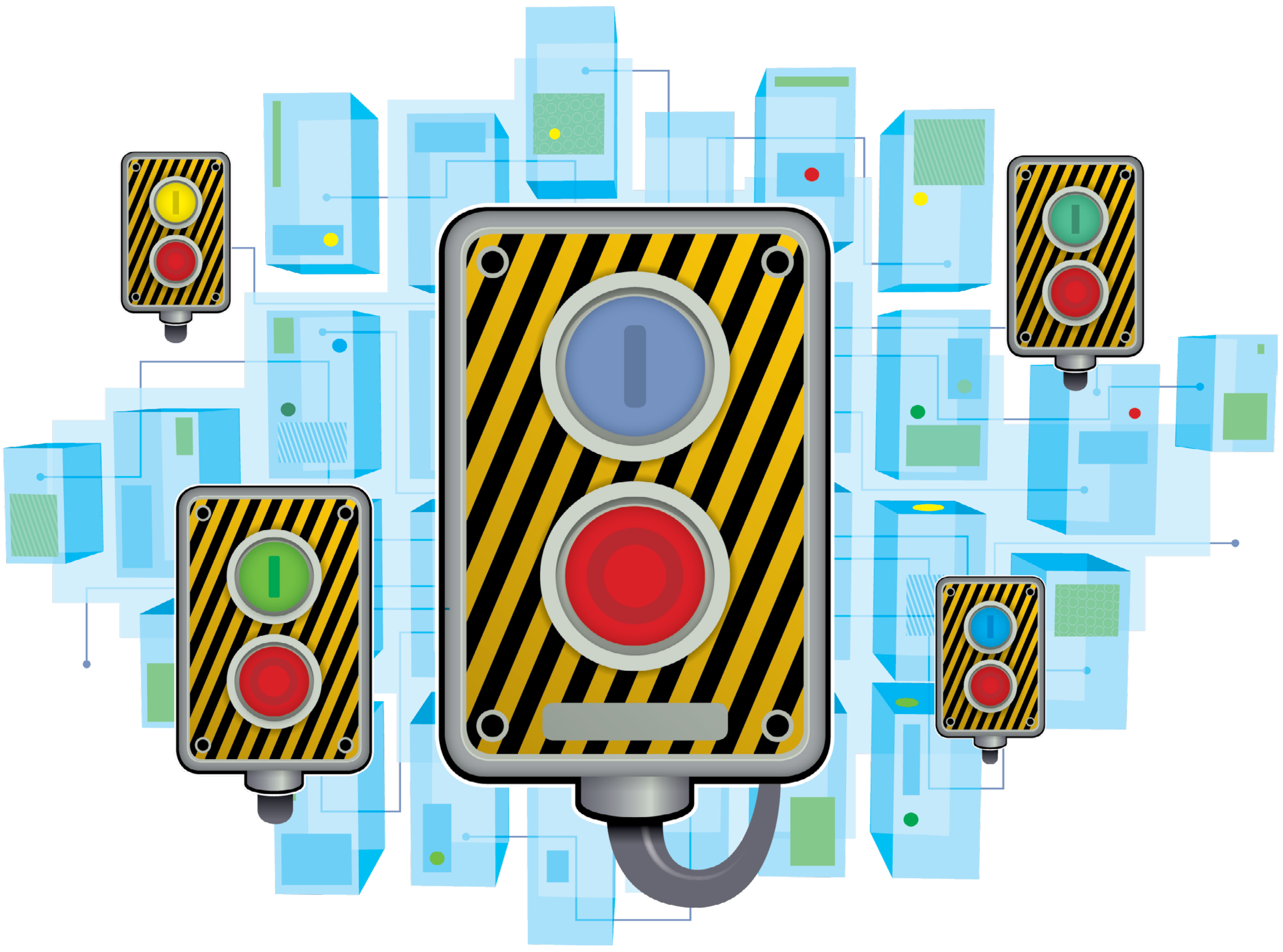

What on Earth is Open vSwitch?

Marco Fioretti explains the whys and wherefores of the mechanism that enables Linux to support swarms of restless virtual servers.

Q Let's dive in then – what on Earth is Open vSwitch?

A Open vSwitch (**http://openvswitch.org**) is open source software that works as a virtual Ethernet switch (or bridge, since in networking the two terms are synonyms). Virtual switches forward traffic between different virtual computers on the same or on different hosts and networks.

Q I assume that this because virtual computers can't use real Ethernet switches directly?

A That's correct. Virtualisation makes one physical computer behave, as far as the operating systems simultaneously running on it are concerned, as if each operating system had one whole, real computer all by itself. The actual software that is playing the trick is called the hypervisor. Among other things, hypervisors need internal switches to route packets to and from those virtual machines as if each of them had its own, independent Ethernet connection(s).

Q *VirtualBox* and similar programs could already do that. What was wrong with their previous ways to handle packet switching, whatever they were?

A The fact is that they couldn't really handle the huge, multi-server virtualisation deployments that are required today.

Q You mean the companies that are pushing all those on demand and cloud computing services?

A Exactly. In order for those services to remain affordable, while providing a decent level of services to an ever-larger market, the servers and networks used by those businesses need virtualised environments with much higher performances and flexibility. They need to be better than somebody running two or three operating systems in parallel on their own office or home computer.

Q Why? Of course, they will have much higher traffic and processor loads, but what is the qualitative difference?

A These days, it looks like everybody and his cousin needs a virtual server in the cloud. Sometimes it is for new projects big enough to need much more processing power, but too short to justify the purchase of new hardware. Much more often, it's just for a few days or hours, to test some concept or just for the fun of it. Besides, many traditional activities that are 'moving to the cloud' have natural seasonal peaks (think to tax accounting or Christmas online sales). On a large scale, this situation literally creates swarms of virtual servers that change all the time, jumping like kangaroos from one server farm to another, in order to stay closer to their end users, or just to distribute the load.

Q Sounds awfully complicated. So, Open vSwitch (let's call it OvS) was developed to cope with the chaos of virtualisation?

A Yes, that's its goal. The code is specifically written both to handle highly dynamic, unpredictable loads, and to automate as much as possible their reconfiguration and management, including automatic start, restart and migration of many virtual computers without compromising their security, or unnecessarily exposing their traffic.

Q Before looking inside OvS, let me ask one thing: is it based on reusable, open standards?

A Yes, it is. The two main ones are called OpenFlow (**www.openflow.org**) and OVSDB. OpenFlow provides methods to interconnect switches and other networking devices, with streams of packets, unsurprisingly called 'flows'. Each flow can have multiple priorities, routing and filtering criteria expressed in tables.

Q That's all I need or want to know about OpenFlow, I think. What about OvSDB?

A The Open vSwitch DataBase management protocol specifies how to create a database of the switch ports of a network, and how to control them remotely. It is the combination of OvS and OvSDB that allows management software to know what happens in a network of moving virtual machines, and reconfigure them more or less in real time.

Q You said OpenFlow and OvSDB are just the main standards supported by OvS.

A True. There are at least two other features without which OvS couldn't do much: Virtual Local Area Networks (VLANs) partition one physical (or virtual) packet switch into multiple virtual ones. This is necessary, for example, when you have different groups of virtual machines on the same host, but all the traffic of each group must remain completely invisible to machines of the other groups.

VLANs are used in tandem with Generic Routing Encapsulation (GRE, **http://bit.ly/13FtVRd**), a tunnelling protocol developed by Cisco Systems. OvS uses GRE to create point to point links between virtual machines in different data centres.

Q In practice, what would you say are the main features of OvS?

A Well, the first one is automated, centralised management. After that I would mention Quality of Service (QoS), monitoring, and hardware integration.

Q Quality of Service? What does that actually cover?

A QoS consists of guarantees that certain parameter and performances of each traffic flow will remain within desired, predefined limits. A business that is an end user of OvS may, for example, be willing to pay to be sure that the average bandwidth of its virtual servers will never go below X megabits per second, and that they will be allowed to use twice that bandwidth at least Y times a day, for no less than Z minutes each time.

Q Cool. What about the automated, centralised management?

A That is the core of OvS. Through the protocols that I've just explained, OvS allows the administrators of virtual machines, scattered across many different data centres, to model in great detail the topology and current status of their networks. Building on that, OvS also provides support for accounting and the automatic – or, at least, semi-automatic – management of both slow and fast network states.

"The code is specifically written to handle highly dynamic, unpredictable loads."

Q Slow and fast network states? What are they now?

A Slow state is, more or less, a fancy way to call the configuration of a virtual machine, and how it may change over time, if you move it to a less loaded host. The fast state of the same machine would be – simplifying a lot! – what that machine is actually doing in any given moment. That is the processes it is running, and its current connections with the rest of the internet.

Q How does OvS distinguish the virtual computers it is managing and their traffic flows?

A First, by appending tags, which also contain unique names for virtual machines, to all the network packets under its control. Second, by providing methods to perform mass renaming, updates and migration of those tags remotely.

Q Wait a second! So far, you've been talking only of virtual servers migrating from one fixed computer to another fixed one. What if the virtual servers themselves are running on mobile devices?

A Ouch! That's a separate, more complicated issue. At the end of the day, what OvS does is 'simply' managing Ethernet frames. As such, it can't handle, all by itself, many cases of physical hosts moving from one access point to another of their own will, so to speak. That, however, is a general problem: many wireless hotspots block all packets with a source MAC address different from the one used to establish the connection, regardless of which virtual switch was used.

Q The only main feature you haven't explained yet is hardware integration.

A This means two symmetrical things. To begin with, OvS can offload parts of the packet flow processing to hardware chipsets, to increase overall performances. The opposite is also true, and is a further, even greater advantage of hardware integration: usage of switches and network cards that understand OvS flows and commands means that you can manage both virtual and physical switches from one interface.

Q What are the main components of the OvS distribution?

A The most important is the daemon that performs the actual switching job. Then there is a database server storing the OvS configuration and a Linux kernel module that supports flow-based switching. In addition to all this, there are utilities for monitoring and debug, plus libraries to write extensions.

Q Last question: what's the status of OvS support in Linux?

A OvS is included in the Linux kernel from version 3.3, with binary packages available for the most popular distributions. In general, OvS can work on any Linux-based virtualisation platform running at least version 2.6.18 of the kernel. Extra support is present in Citrix XenServer and Red Hat Enterprise Linux. Linux aside, OvS has also been ported to FreeBSD, Windows and some embedded systems. LXF