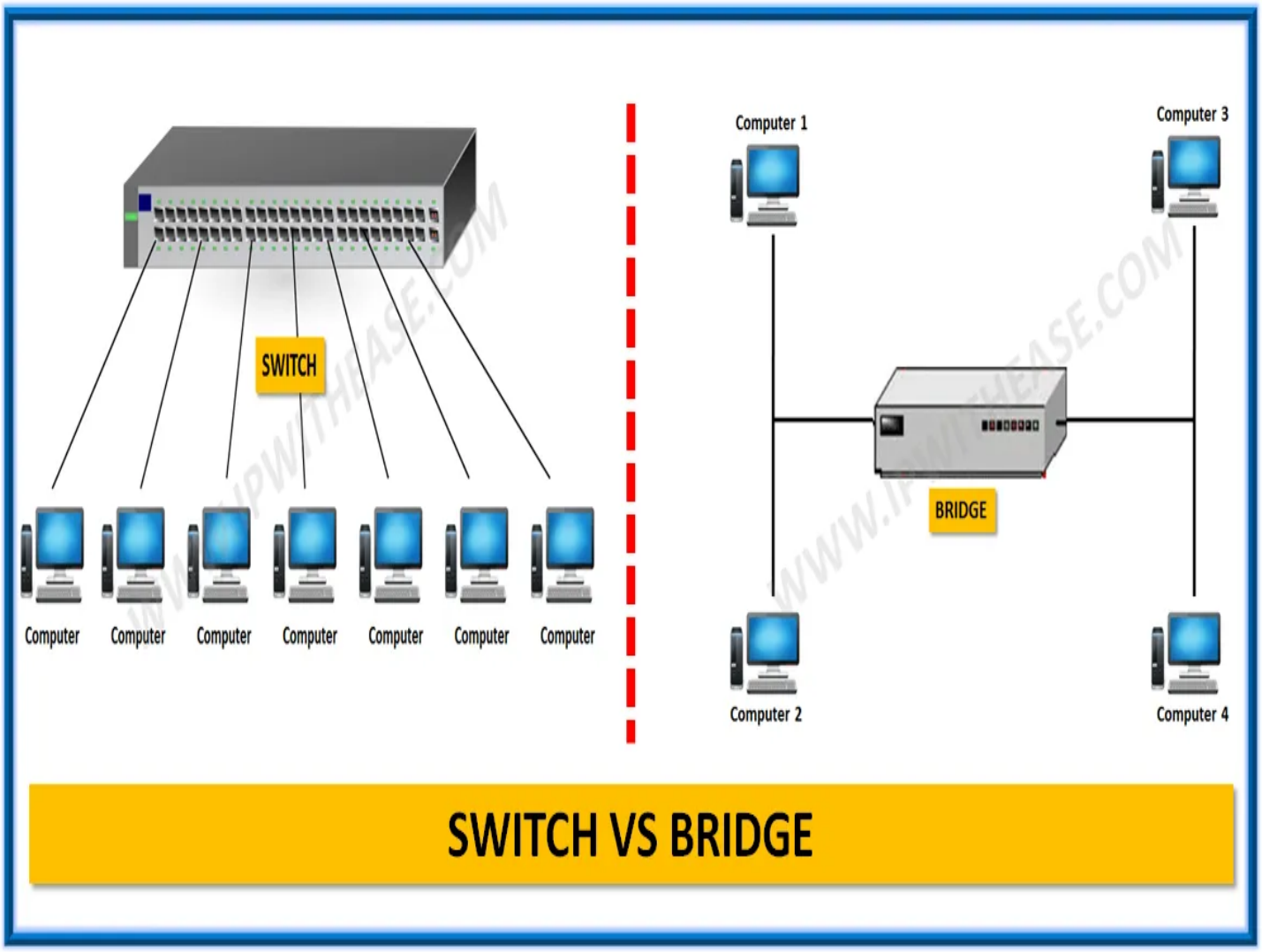
Bridge vs. Switch: What I Learned From a Data Center Tour

March 28, 2021 (Updated: August 4, 2021) • Networking

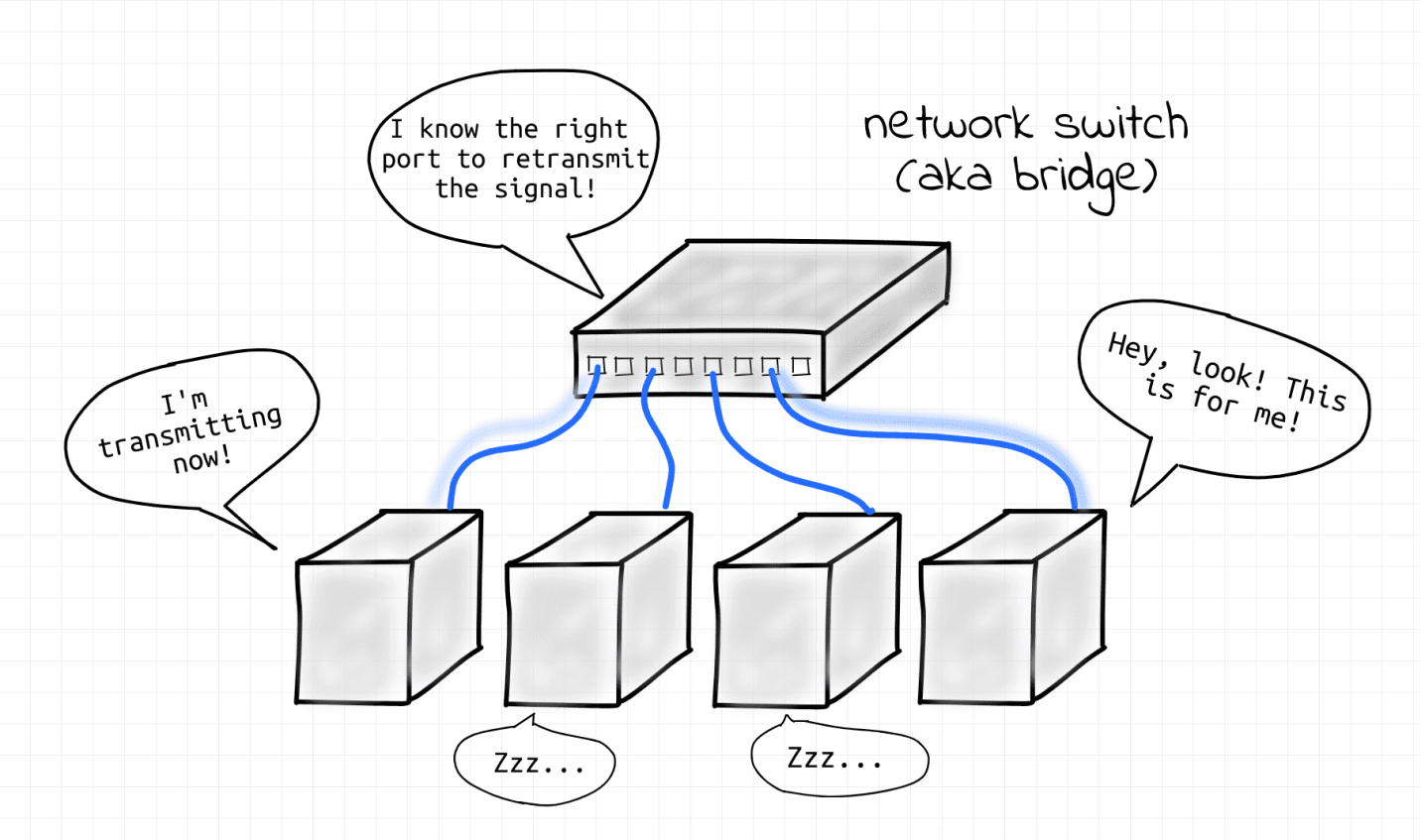


Many thanks to Victor Nagoryanskii for helping me with the materials for this article.

The difference between these two networking devices has been an unsolvable mystery to me for quite some time. For a while, I used to use the words ***"bridge"*** and ***"switch"***

interchangeably. But after getting more into networking, I started noticing that some people tend to see them as rather different devices... So, maybe I've been totally wrong? Maybe saying *"bridge aka switch"* is way too inaccurate?

Let's try to figure it out!



Switch == Bridge

There is a nice book called "Understanding Linux Network Internals". It has a whole chapter on bridges. Among other things, it states that ***a bridge is the same as a switch***.

Bridges Versus Switches

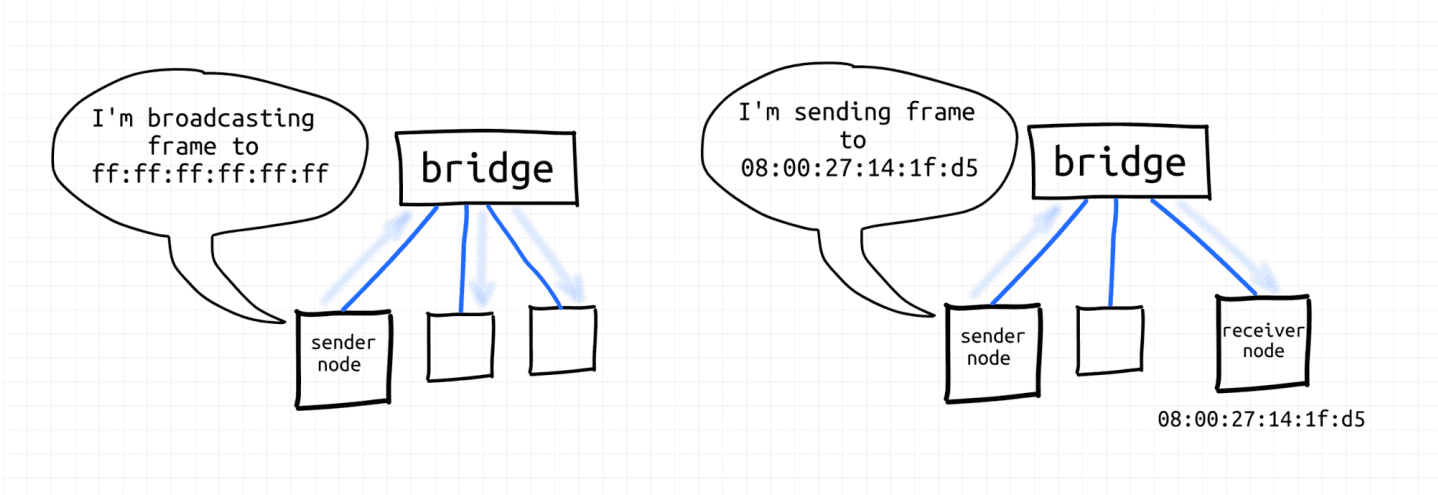
> The terms *bridge* and *switch* can be used to refer to the same device. However, nowadays the term *bridge* is mainly used in the documentation (such as the IEEE specifications referenced at the end of this chapter) that discusses how a bridge behaves and how the STP (which we will see in the next chapter) works. In contrast, references to the actual devices are usually made with the term *switch*.
>
> The only cases where I have seen people referring to a bridge device using the term *bridge* is when the device is equipped with only two ports (and bridges with two ports are not that common nowadays). This is why I often define a switch informally as a multiport bridge. Unless you are familiar with the official IEEE documentation, you will probably use the term *switch*. I personally worked on bridging software for years, and as far as I can remember, I used the term *bridge* only when working on the documentation, never to refer to a device on any network setup.
>
> Generally speaking, I can say that there is really no difference between a bridge and a switch.
>
> *Quote from the "Understanding Linux Network Internals" book*

My take from this book was that:

- By saying "bridge" we refer to a *logical function* of a device.
- By saying "switch" we refer to the actual physical device performing that function.

So, what is this *function*?

*Bridges **transparently** combine network nodes into Layer 2 segments creating Layer 2 broadcast domains. Nodes of a single segment can exchange data link layer frames with each other using either unicast (MAC) or broadcast addresses.*



Bridge network function.

> Bridges can combine not just end-nodes, but also sub-segments. I.e., one can connect a bridge to a bridge, for instance, doubling the max segment size.

> Bridges perform their function transparently. From the network participants' standpoint, bridges don't exist. Nodes just send frames to each other. The task of the bridge is to learn which node is reachable from which port and forward the frames accordingly.

Switch != Bridge

This finding reassured me for a moment, so I felt brave enough to use the *"bridge aka switch"* statement in my recent write-up on low-level networking fundamentals. I was very lucky the day I published it, so the article got quite some attention and feedback. And some of it was about my bridge-switch assumption. Apparently, not everyone was happy about it…

But you know why I absolutely love the Learning in Public idea? Because of the thing that happened next. A practicing Network Engineer reached out to me and kindly offered a virtual excursion into a data center. That sounded like an awesome opportunity for me to take a look at the true enterprise-grade network and ask some questions to a real networking guru.

Below are my learnings from that trip.

Historical discourse

Seems like there is a long history behind the evolution of networking devices. Supposedly, bridges started as two-port devices combining two L2 (or L1?) network segments into a bigger L2 segment. Hence, the name *bridge*. Then they turned into multi-port devices, but the name stayed. Regardless of the number of the ports such devices were unnoticeable from the network participant standpoint - bridges always perform their function transparently.

Then, the hardware evolved even further and nowadays, it's hard to find pure hardware bridges, especially in a serious context of use (see the next section). However, the need for the logical bridge function stayed unchanged. In the real world, this function is often performed by *switches*. In the virtual world - by *Linux bridge* virtual device.

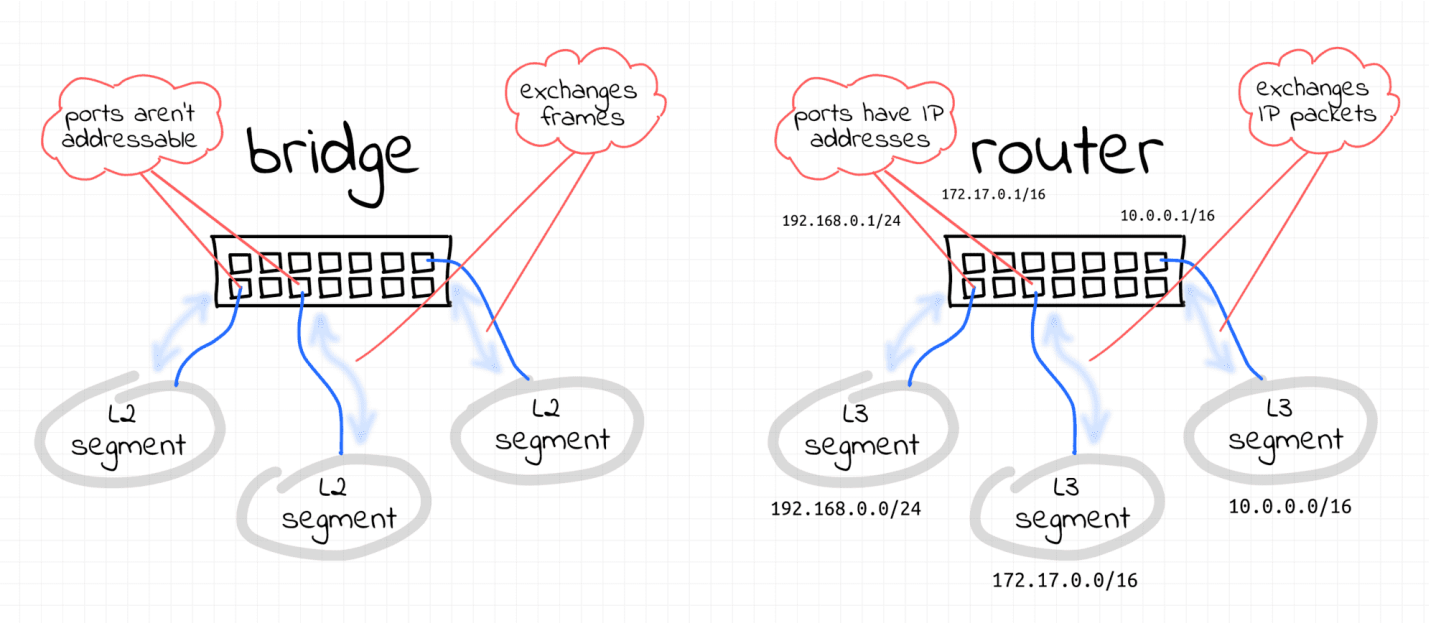
In the physical world, a bridge connects roads on separate sides of a river or railroad tracks. In the technical world, bridges connect two physical network segments. Each network bridge keeps track of the MAC addresses on the network attached to each of its interfaces. When network traffic arrives at the bridge and its target address is local to that side of the bridge, the bridge filters that Ethernet frame, so it stays on the local side of the bridge only.

If the bridge is unable to find the target address on the side that received the traffic, it forwards the frame across the bridge, hoping the destination will be on the other network segment. At times, there are multiple bridges to cross to get to the destination system.

Switch >= Bridge

What helped me to understand the difference is that I started thinking in terms of the following logical networking devices:

- **Bridge** - transparent Layer 2 device performing frame forwarding on a single L2 segment.
- **Router** - non-transparent Layer 3 device performing IP packet forwarding between multiple L3 segments.



So, how a reasonably big LAN can be organized? Theoretically, it's possible to interconnect multiple bridges to extend a broadcast domain up to hundreds (or even thousands) of nodes.

But, it turned out that giant broadcast domains are pretty hard to manage. Apparently, in the real world, huge domains often lead to huge outages.

Thus, instead of having an immense data-center-wide broadcast domain, it's better to have smaller isolated L2 segments that can talk to each other through... L3 routers!

Getting, back to switches...

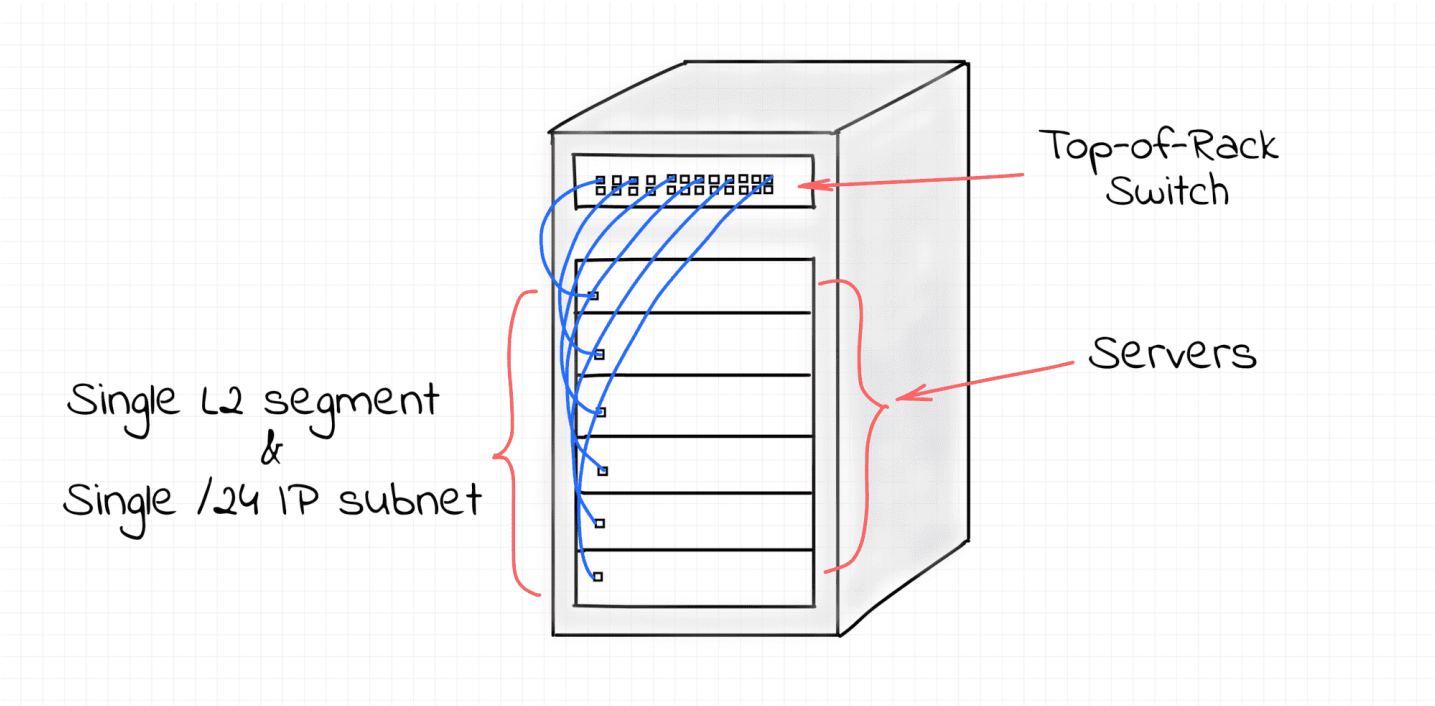
Modern network switches used in data centers are pretty advanced devices. They can work as bridges. Or, as routers. Or... some of their ports can work in a bridge mode, and some others - in a router mode. I'd speculate a bit and say that it's likely we're talking here about *multilayer switches*.

In a setup I was introduced to, all servers on a single rack belong to the same */24 IP subnet* and are connected to a single switch.

> Facebook seems to have a pretty similar DC networking architecture. I didn't research much, but this Load Balancer article indirectly confirms it.

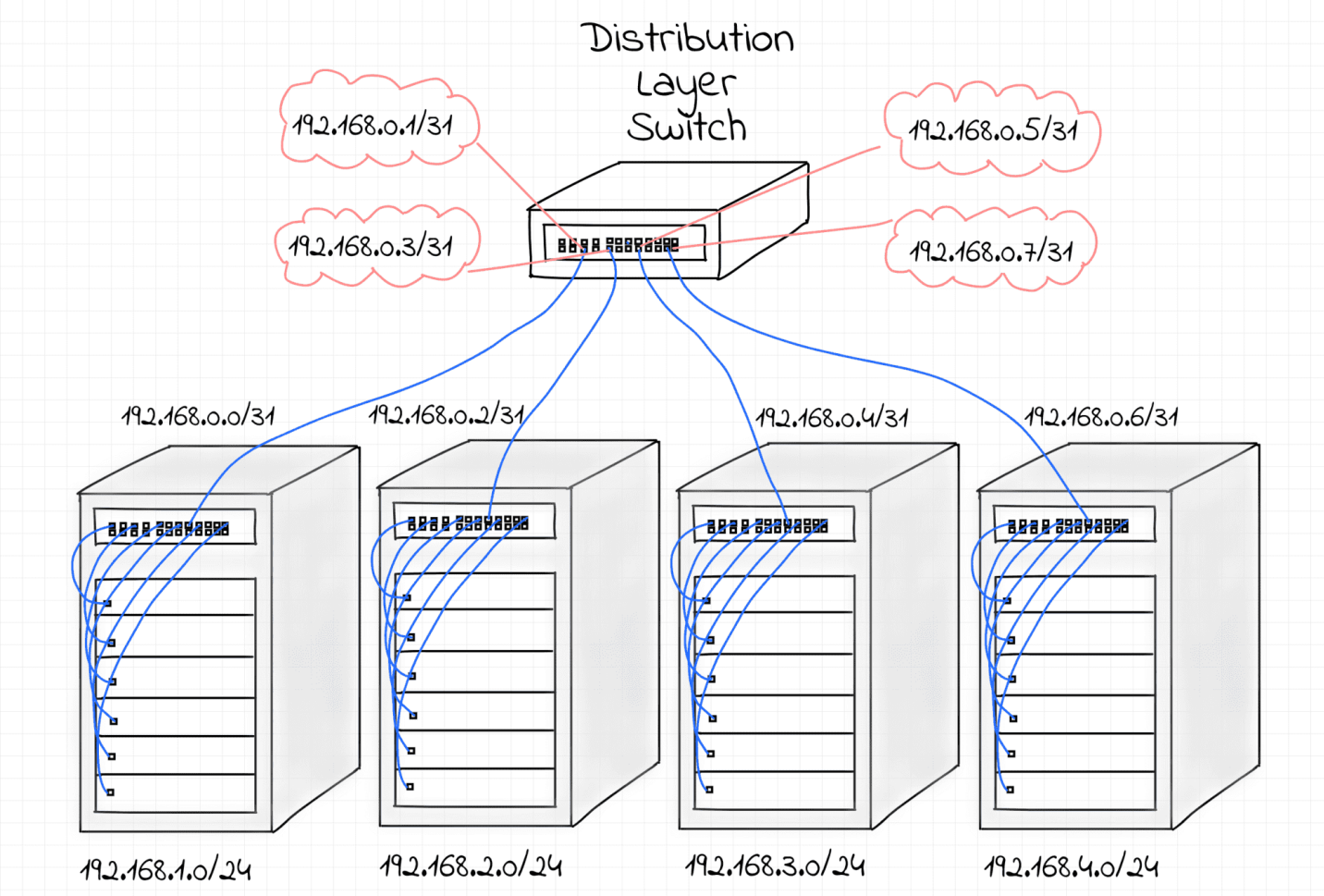
Such a switch is called *top-of-rack (TOR)*.

For these servers, the TOR switch behaves as a canonical transparent [multi-port] bridge. I.e. as a pure Layer 2 device. And a rack forms equally sized L2 and L3 segments.



However, the L3 segments formed by individual racks need to be joined into internetwork. For that, one of the remaining ports of the TOR switch is configured to work in L3 mode. Unlike the

L2 ports, it means that this port is an addressable network node with an IP address. It's then connected to a higher-layer switch.



Simplified hierarchical internetworking model.

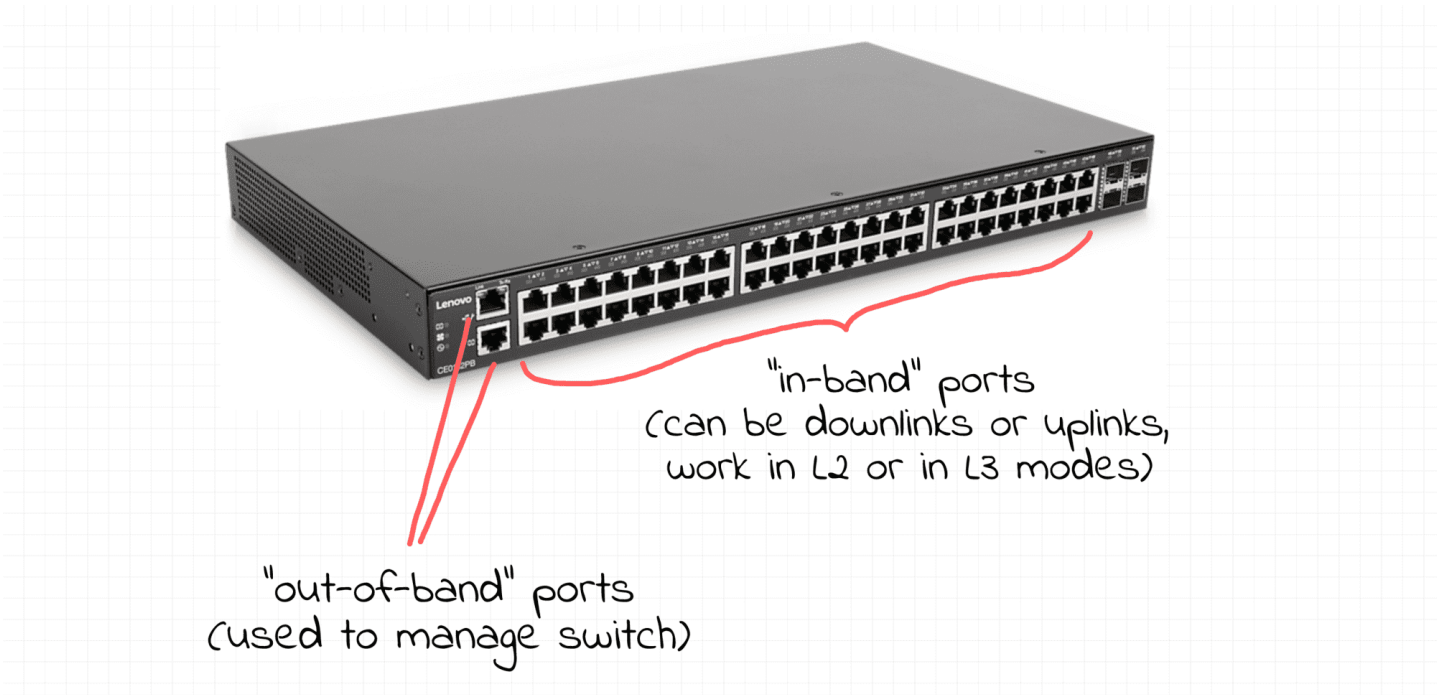
Disclaimer: actually, every TOR switch is connected to at least two higher-layer switches. First of all, to provide some redundancy in the case of hardware failure. But also if these multiple physical connections are combined into a single logical link, it can increase the resulting throughput. However, for the sake of simplicity, let's omit this part.

These *higher-layer switches* on the diagram above are called *distribution layer* switches. In the network I was looking at, switches on this layer work as pure L3 routers. Hence, each port of a distribution-layer switch is a full-fledged Layer 3 device with an IP address assigned to it.

TOR-switches can be thought of as compound devices with a bridge (with tens of ports) and a router (with just 2 ports) inside. Distribution layer switches can be thought of as multi-port L3 routers. Like truly *multi*-port. 48- or 64-port routers! The router on a TOR-switch knows only two routes - to the rack's subnet and the default one, pointing to its distribution layer switch. And every distribution layer switch knows a lot of routes. Every rack connected to it has its own /24 IP subnet and this switch works as a border router for tens such subnets.

However, physically, there is no difference between the first and the second types of switches! They all look the same, but just configured differently.

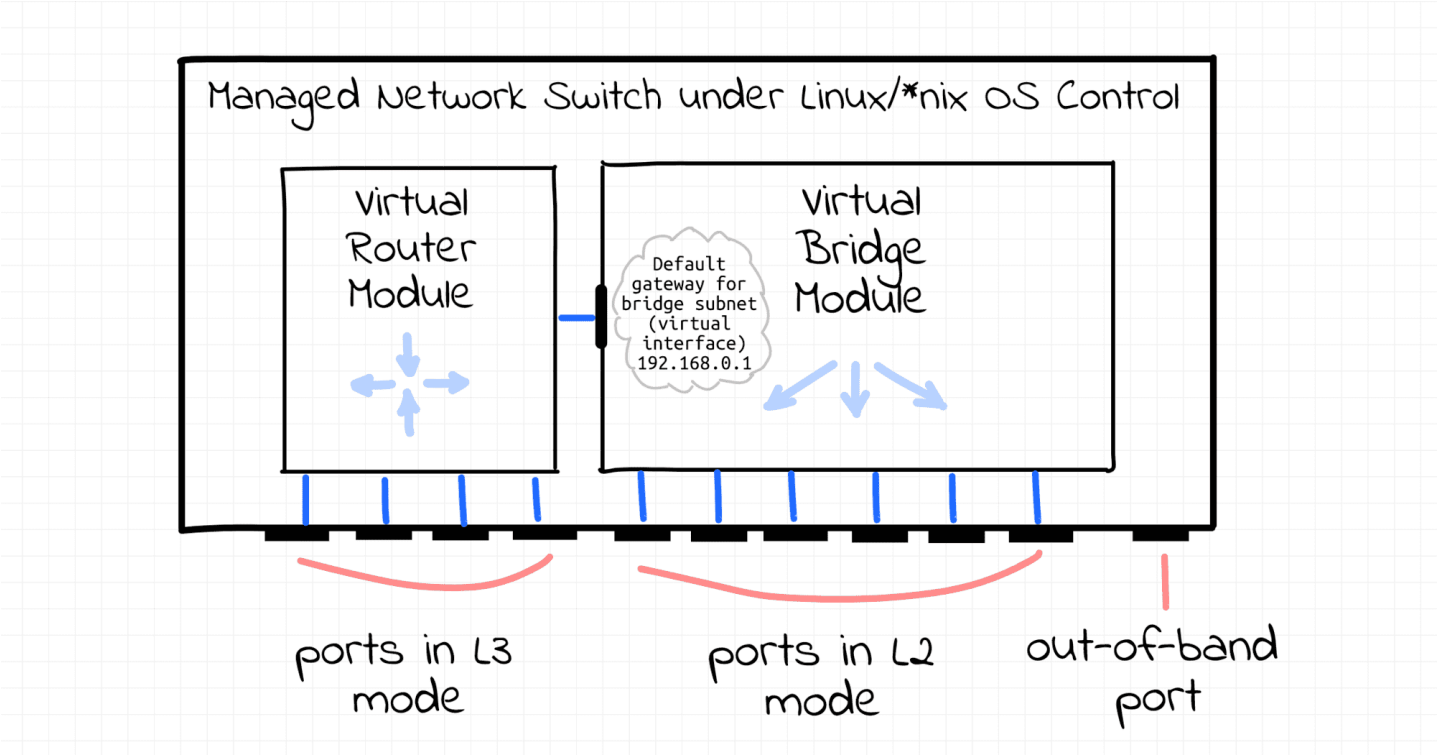
In addition to these 48 (or 64) ports, switches have one or two *out-of-band* ports.



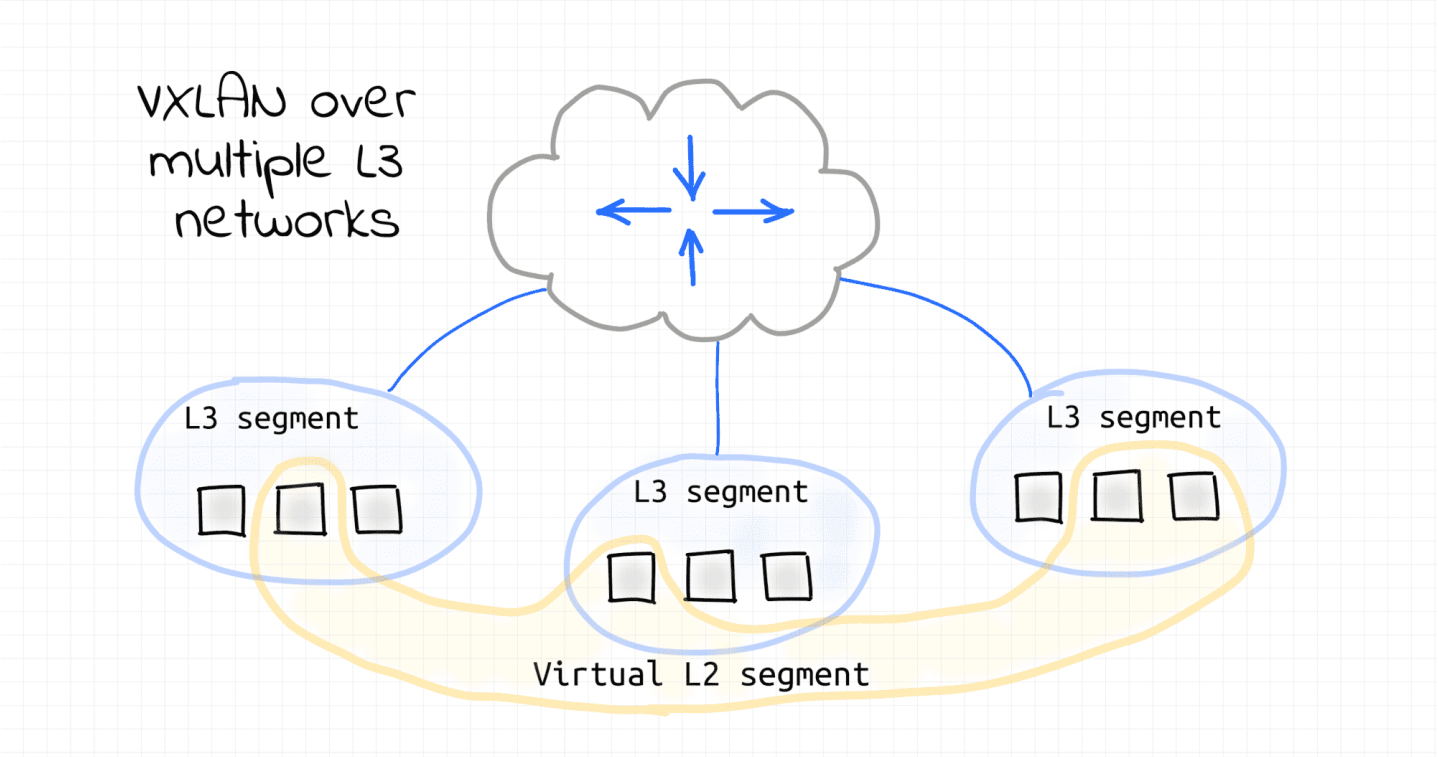
Regardless of the mode of other ports, the *out-of-band* ports always work in L3 mode. One can log in on a switch using an IP address of an *out-of-band* port. This is needed to manage switches because configuring switches through the normal ports would be simply dangerous. Imagine, you messed up with some commands and blocked yourself out of the switch?

Curious what would it look like when you *ssh* to a switch? Surprise, surprise! It's Linux! Or FreeBSD. Or a proprietary Unix-like OS. I.e. one can configure a switch via a traditional ssh session using widely-known *iproute2* tools such as `ip` and/or `bridge`.

From the management standpoint, every port on a switch looks like a traditional network device. One can enslave some ports to a *virtual Linux bridge*, assign some other ports IP addresses, configure packet forwarding between ports, set a route table, etc. So, you can think of a switch as of a Linux server with many-many network ports. And it's totally up to you how to configure them. All the traditional Linux capabilities are at your disposal. But of course, from the hardware standpoint, switches are highly-optimized packet processing devices.



So, what's up with broadcast domains? Can we have a broadcast domain spanning multiple racks? Sure! VXLAN to the rescue! Up until this point, I was describing a physical network setup of a data center. But one can configure any sort of an overlay network on top of it making it tailored for the end-users use cases.



Instead of conclusion

Actually, it was quite exciting for me to see the real production network in action. Things that I learned on that journey make a lot of sense to me. However, I'm pretty sure that it's not the only

possible setup. So, I'd appreciate it if you share your experience here or on twitter.

Have fun!