

Notes on Tunnels

1 Introduction: The ideal network

When learning about computer networks, you will have been introduced to a wide array of concepts. You will have learned about encapsulation and decapsulation, the OSI- or TCP/IP layer models, and about the different network nodes that make a network. In the example systems that typically are presented in networking courses, all actors are playing by the same rules, being configured with the same routing protocols, allowing for (somewhat) unproblematic networking. Assuming everyone plays by the same rules does not align with the real world of networking however. Not all organizations want to deploy the same hardware, which might change what networking protocols they have available. Some organizations might use a whole different IP version, which leads to incompatibility between networks [1]. Even if they did follow the same network design rules, the “normal” network model unfortunately does not scale efficiently [2]. Networks need to be adapted to their context and requirements, in the end leading to a heterogeneous mass of designs and deployments.



Figure 1: Directions can be confusing.

The lack of uniform, predictable and scalable networks creates a need to adopt techniques that circumvent the problems or enhance the functionality of the network. One such technique, is the use of tunnels. Tunnels in networks are similar to tunnels on roads: you go in on one end and come out somewhere else. For some complex tunnel systems, you might not even know where you will end up, but just follow through a tunnel.

In the case of network tunnels, the packets are the travellers that pop out in different locations. Before tunnels can be utilized, however, they must be created. This document is intended to present some of the motivation for, design architectures of and techniques to deploy network tunnels.

2 What is a tunnel?

A tunnel is a logical communication path between tunnel endpoints, implemented by carrying packets of one protocol inside another. A collection of tunnels and tunnel endpoints can form an overlay network. Often times, when using an overlay network the underlying transport network is opaque for the tunnel clients/users. From our road tunnel analogy, the transport network is the mountain or underground that our tunnel is traversing, while the overlay network endpoints are the entrance and exit of the tunnel.

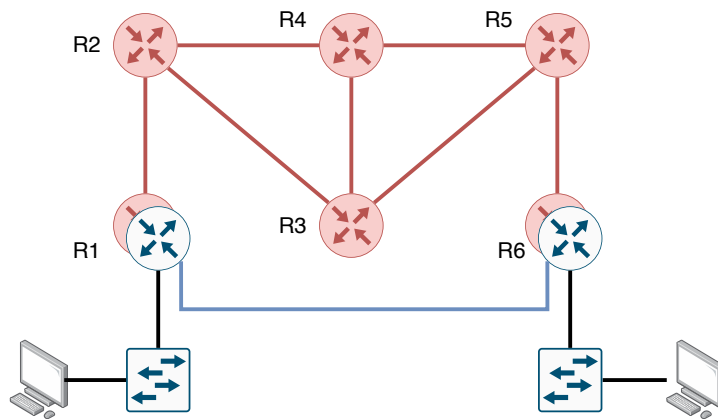


Figure 2: Example overlay network. R1 and R6 have an active tunnel that

Figure 2 shows an example of how a tunnel could be deployed. In the figure, R1 and R6 have a tunnel that creates a logical link between them. If this tunnel is the preferred route between the computers, the computers would not have visibility over what was going on in the “red” network. When performing a traceroute from one computer to the other, the intermediate underlay routers will normally not appear as ordinary overlay hops. The tunnel has in essence created an overlay or a masking of the underlying transport network, which allows for a simplified network view from the hosts’ perspective.

Realizing the tunnel can be done in a number of ways, but in broad terms is done through adding a protocol header to the packet. Which header is used will be further presented in Section 5.

3 Why would I build a tunnel?

As presented briefly in Section 1, tunnels may be a necessary tool to provide connectivity between networks. The reasons for employing a tunnel can vary widely depending on the perspective.

From the perspective of a network provider, you may want to employ tunnels at your ingress and egress points of your network. By, for instance, deploying tunnels at the edges of your network, you may circumvent the need for BGP-routes being distributed into your

core network, leading to a BGP-free core [3]. Having a BGP-free core reduces the memory needs of core network components, which can both save money (TCAM/CAM is expensive), electricity or improve performance as forwarding capacity can be prioritized.

From the perspective of a network consumer tunnels may also be attractive. A business may need to connect multiple locations into a virtual private network (VPN), establishing connectivity for systems in different geographical areas. Often times, an organization utilizes private network address spaces in their networks, addresses that typically cannot be routed over the internet. Tunnels can solve this problem and are often times the building-blocks of VPNs. The tunnel creates an overlay, where the opposing private network is presented as an almost locally available destination network, through a set of tunnelling routers. Figure 2 shows an example of this type of overlay, where R1 and R6 act as tunnelling routers and the “blue” network could be created using private network addresses.

Another use for tunnels is to apply network policies and mechanisms to all traffic heading to a destination. This is in essence what a consumer VPN does as the intended network policy is encryption of traffic, while the destination in a consumer VPN typically is a VPN-server hosted by the VPN provider. In tunnels where an organization controls both end-points of the tunnel, encryption can be applied so that the traffic over the transport network is encrypted. Although many applications (layer 5) automatically natively deploy encryption on user data (HTTPS, for instance), there are applications where this is not the case. Utilizing network encryption is relevant in such use-cases especially for organization’s internal networking traffic if this is transported by a network provider.

4 Where should my tunnel go?

We’ve already introduced one design of a tunnel architecture in Figure 2 and Section 3, but there are other useful architectures to introduce. Firstly, the site-to-site architecture we’ve referred to so far is a special case of the more general point-to-point tunnel. Placing the tunnel on a router and utilizing it for all the traffic from a site is already an enhancement of a tunnel created between two end-point systems. Creating a tunnel between two hosts can be useful when applying encryption or other network policies to traffic destined for a specific endpoint. The site-to-site architecture is just the extension of the point-to-point architecture, increasing the number of “tunnel clients” utilizing the tunnel. It is also possible to have a point-to-site architecture, where a single host connects to e.g. a gateway router, that has the business network connected behind it. The point-to-point, point-to-site or site-to-site architectures also act as the building blocks for the other tunnelling architectures we will introduce. Figure 3 shows simplified illustrations of the different tunnel types, with the blue components/links being part of the overlay network

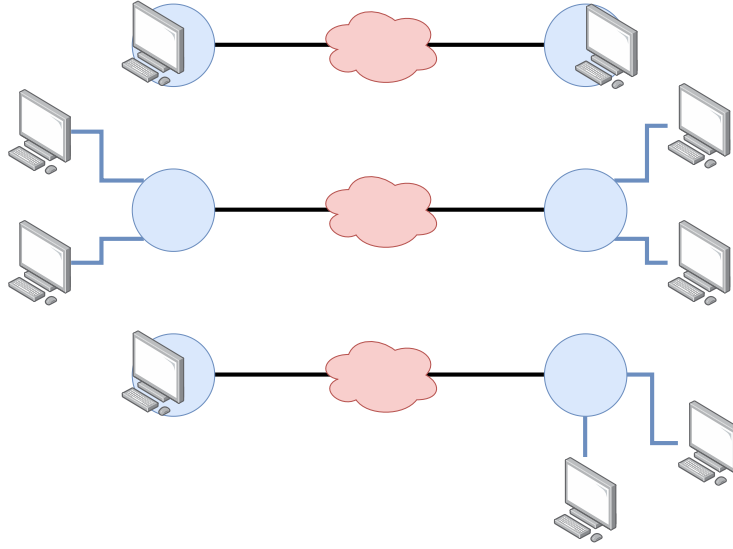


Figure 3: Point-to-point, site-to-site and point-to-site tunnels respectively.

4.1 Hub and Spoke

In the context of a consumer VPN, we stumble on a different architecture. From each of the phones or devices that are connected to the VPN-service, a point-to-point tunnel is established to the VPN server, but the server likely serves multiple customers. This design is called a hub-and-spoke architecture, where multiple spoke nodes connect to the same logically central hub, illustrated in Figure 4. Hub-and-spoke tunnel topologies are also relevant for organizations with one central office and multiple remote offices/users. The central office can act as the central tunnelling end-point, where remote users or offices can connect to, for example, the business network. In instances where spoke nodes need to communicate, the central office can still be relevant to apply network security mechanisms and policies on traffic travelling between spokes. The hub-and-spoke architecture is efficient and scales well in terms of number of tunnels and complexity at the spoke nodes, as they only need to be configured with one tunnel. The total number of tunnels in a hub-and-spoke system can be defined as:

$$N_{Tunnels} = N_{SpokeNodes} \quad (1)$$

A potential weakness of a hub-and-spoke architecture is the reliance on the hub. The hub acts as a single-point-of-failure in this architecture, and if it fails, all spoke nodes lose connectivity. The hub node also needs to serve the connectivity needs of all spoke nodes and may become a performance bottleneck. There is an additional performance cost in hub-and-spoke networks where two spokes are required to communicate, as the traffic does not take the most efficient route. In instances where traffic has to go between two closely geographically located spokes, the traffic first has to traverse through the hub, which may be located at a completely different

location. The traffic in this instance doesn't just have to traverse from the source spoke to the hub, but also has to take the return journey to the destination spoke.

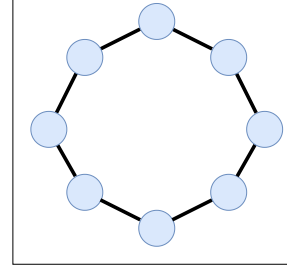
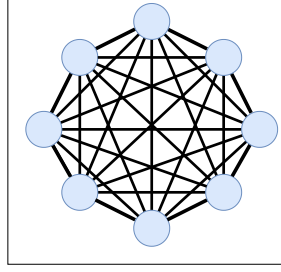
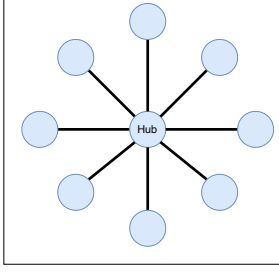


Figure 4: A hub-and-spoke architecture. Figure 5: A full mesh architecture. Figure 6: A ring architecture.

4.2 Full Mesh

An alternative tunnelling architecture, that avoids the single-point-of-failure of the hub-and-spoke design is the mesh or full-mesh architecture. In this design, all the tunnelling nodes create a separate tunnel to each other tunnelling node, as shown in Figure 5. This design has benefits over the hub-and-spoke design as the hub no longer is required to process the traffic of all the spokes. Instead, traffic is only processed by the concerned parties in the communication, leading to a more efficient use of the data-plane. Traffic can be sent directly to the intended tunnel endpoint without traversing an intermediate overlay hub, although the actual underlay path may still be indirect. The full-mesh design does, however, require a more complex control plane and may require a great deal of configuration overhead to properly deploy. The full-mesh topology scales quadratically in number of tunnels, as the number can be represented by:

$$N_{Tunnels} = \frac{N_{Nodes}(N_{Nodes} - 1)}{2} \quad (2)$$

To manually configure this number of tunnels requires laborious effort with only a relatively small number of nodes. There are services that assist in the creation of such topologies in the consumer market, notable examples being Tailscale or ZeroTier. These providers typically market their service as a private network of some form, but are in essence services supplying a full-mesh overlay network with encryption between your devices.

4.3 The Ring

Another alternate topology, rarely used as an explicit tunnel topology, is a ring-architecture, where each node communicates with two other nodes, as shown in Figure 6. This architecture alleviates the high amount of tunnels from the full-mesh and the single-point-of-failure

problems of the hub-and-spoke designs, but brings its own challenges. Firstly, designing an efficient ring topology (which nodes speak to which) can be a complex design problem. Should the connections be defined by geographical proximity or should they be defined by which end-points communicate most frequently with each other? There is no correct solution to this problem and it must be taken into consideration on a case-by-case basis. If incorrectly configured, or for topologies where this cannot be configured efficiently, traffic needs to be processed by multiple nodes, leading to a processing overhead in the network. The number of tunnels does however scale very efficiently, as it scales linearly with the number of nodes included in the network:

$$N_{Tunnels} = N_{Nodes} \quad (3)$$

4.4 The Best Architecture

The most appropriate architecture must be considered on a case-by-case basis as no single tunnelling architecture is best in all situations. Sometimes a combination of architectures makes more sense, bringing in positives (and negatives) from the different topologies. A mesh-architecture doesn't necessarily have to be configured in full-mesh, but can have a fixed number of peer nodes. If you deconstruct such a "non-full-mesh" you may see that it in principle becomes similar to a ring-topology, just with a mesh of rings. Another combination topology could for instance be the combination of the hub-and-spoke- and ring-topologies. By setting up a hub-and-spoke architecture with a backup ring-tunnel you may alleviate the hub-and-spokes reliance on a single-point-of-failure while not requiring unmanageable configuration overhead.

In practice, tunnelling architectures may become messy and complicated when the overlay networks grow, similar to how the tunnel-architectures become complicated meshes in physical systems, as shown in Figure 7.

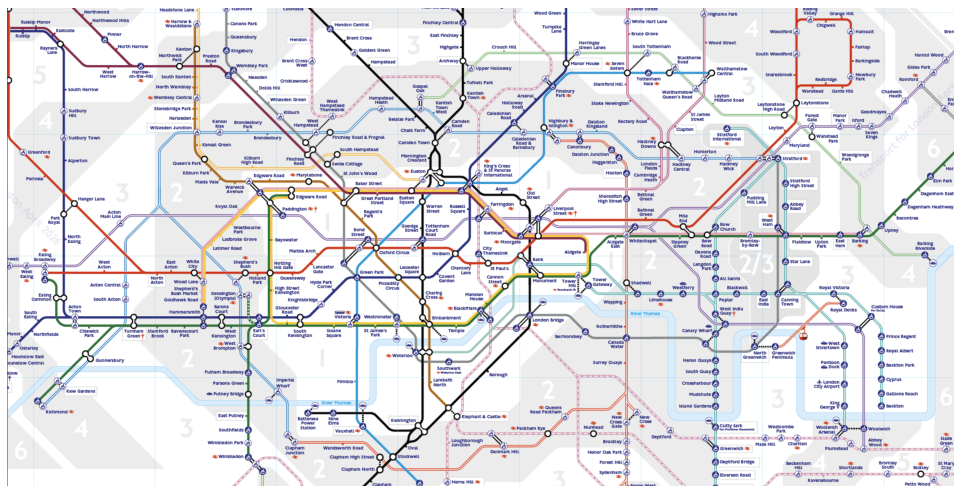


Figure 7: The London Tube system is a set of many tunnels where you might need to go through multiple hub-stations when travelling between distant stations (spokes).

5 How can I build my tunnel?

To end with even more confusion on an already confusing topic, we move on to how to technically deploy a tunnel. As previously alluded to, the protocols used to deploy tunnels are not fixed and must be chosen based on the requirements of each use-case. In some use-cases the only requirement is encryption to avoid actors in open/public networks from eavesdropping on your data. In other use-cases, systems may require Layer-2 connectivity, which can be virtualized through a tunnel. To our knowledge, no single source appears to provide a comprehensive taxonomy of tunnelling technologies. The most comprehensive one in circulation (albeit often reformatted and lacking citation) is from a 2006 survey from the now non-existent XO Communications, shown in Figure 8.

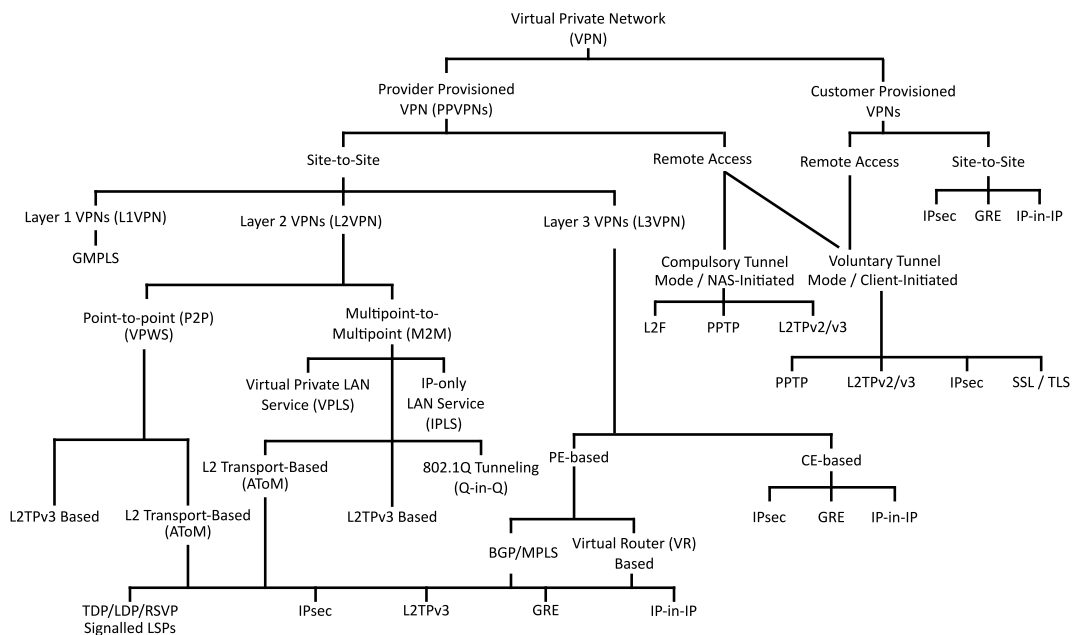


Figure 8: VPN taxonomy from 2006 [4]. Several technologies are missing from this taxonomy, while a number of the listed technologies are legacy and superseded.

We can however try to cover some of the general types and give examples within each of these. Note that each tunnel type may be part of several of the general types

- **Layer-3 Routed IP Tunnels:**

The most widely used consumer tunnels are Layer-3 routed IP tunnels that provide logical Layer-3 connectivity between tunnel endpoints or attached networks. These tunnels encapsulate user-data in extra sets of headers and route the traffic through the use of IP addresses in the transport network. For consumers, these tunnels may be part of VPN-applications, e.g. Tailscale or OpenVPN utilizing e.g. WireGuard as the underlying tunnelling service. These tunnels are also often used in professional contexts for remote access VPNs or site-to-site VPNs, using e.g. GRE [5], or to provide isolations between customers in datacenters [6].

- **Layer 2 Ethernet Tunnels:** In layer 2 tunnels, Ethernet frames are carried across an underlay, making remote Layer-2 segments appear logically adjacent. Layer 2 tunnels are not common in consumer contexts, outside some specific OpenVPN configurations, but are deployed in a variety of business contexts. Inside datacenters or in distributed Kubernetes-clusters layer 2 tunnels provide networking flexibility allowing for cloud-workloads to be done in geographically (or at least network-logically) distributed systems. VXLAN [7] and GRETAP are technologies that are used to deploy layer 2 tunnels.

- **Application Tunnels:**

Somewhat as an outlier from the other protocol types listed are application tunnels. Application tunnels are protocols or solutions that don't tunnel all traffic from a system, but only select traffic. This can for instance be achieved through the use of SSH-port forwarding, where traffic that is sent to local sockets is forwarded to a remote destination through an SSH-tunnel. SOCKS, similarly is another application tunnel, with the added functionality of the remote system being able to function as a proxy for the traffic, allowing for a solution similar to an X-to-site tunnel.

Tunnel technologies can also be distinguished by the additional functions they provide or the deployment context in which they are used:

- **Basic Encapsulation Tunnels:**

These are possibly the simplest form of tunnel, where an extra header is inserted to tunnel the traffic to the tunnel destination. Often times encapsulation tunnels are applied at layer 3, for instance utilizing IPvX-in-IPvX (X as in 4/6) or GRE-tunnels. The tunnels may be used either to establish VPN connectivity in a site-to-site context, but can also be used to solve compatibility requirements when traversing IPv4/6 networks[8]. Encapsulation tunnels do not provide encryption services by default, but GRE-tunnels sometimes use IPsec to provide encryption for the user-data contained within packets. An issue when using IPv4-in-IPv4 or GRE-tunnels is the fragmentation of IP packets [9]. When adding an outer IP header to a packet, the inner IP header becomes part of the data of the outer header. IP packets may be transmitted up to the link or path MTU, commonly 1500 bytes on Ethernet networks. If such a packet is encapsulated with an additional IPv4 header, the resulting outer packet becomes 1520 bytes and no longer fits within a 1500-byte MTU. The resulting oversized packet must either be fragmented, dropped with an MTU-related error, or prevented by mechanisms such as Path MTU Discovery or configuring a smaller tunnel MTU. Fragmentation increases protocol overhead and processing cost because fragments must be forwarded and later reassembled. The problem can become worse when multiple layers of tunnelling are nested. Although the other layer 3 tunnels may have the same issue, some of them are able to optimize and remedy this factor more efficiently as part of their application.

- **Encrypted VPN Protocols:**

The most widely used modern VPN protocols are WireGuard, IPsec/IKEv2 and OpenVPN. They range in complexity, broadness in adoption and cryptographic implementation, but all attempt to address the need for network layer encryption. For many consumer applications, the encryption protocol is hidden as a component of a more complete software solution, but often times, the underlying mechanism will be one of the three listed.

- **Provider VPNs**

Provider VPNs are large-scale deployments supplied by a network operator selling services directly to businesses[10]. Provider VPNs often utilize MPLS to enable their functionality, for instance in Martini-VPNs or MPLS L3/L2VPNs, but can also be realized with Layer-2/3 tunnels, such as VXLAN or IPsec.

Outside the examples listed, there are still relevant tunnelling technologies in use. Cloud providers may have a custom set of tunnels they use for their services. With the adoption of QUIC, new QUIC-based tunnelling mechanisms are emerging. Tor is in essence a privacy-tunnel mechanism. Many more examples of tunnels exist, each filling a particular niche. There is likely a version of a tunnel that fits very specifically and nicely into every use-case for tunnels, but having the full overview of them all is out of scope for this lecture note.

References

- [1] S. Sharwood. (2021, Jul.) China sets goal of running single-stack ipv6 network by 2030, orders upgrade blitz. [Online]. Available: <https://www.theregister.com/on-prem/2021/07/26/china-sets-goal-of-running-single-stack-ipv6-network-by-2030-orders-upgrade-blitz/1157788>
- [2] G. Huston. (2021, Mar.) What will happen when the routing table hits 1024k? [Online]. Available: <https://blog.apnic.net/2021/03/03/what-will-happen-when-the-routing-table-hits-1024k/>
- [3] BGP Help, “Bgp free core,” Feb. 2017, accessed: 2026-09-07. [Online]. Available: <https://www.bgphelp.com/2017/02/12/bgp-free-core/>
- [4] M. Lewis, *Comparing, Designing, and Deploying VPNs*, 1st ed. Indianapolis, IN: Cisco Press, 2006.
- [5] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, “Generic routing encapsulation (gre),” Internet Engineering Task Force, RFC 2784, Mar. 2000. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc2784.html>

- [6] D. Black, J. Hudson, L. Kreeger, M. Lasserre, and T. Narten, “An architecture for data-center network virtualization over layer 3 (nvo3),” Internet Engineering Task Force, RFC 8014, Dec. 2016. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc8014.html>
- [7] M. Mahalingam, D. G. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, “Virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks,” Internet Engineering Task Force, RFC 7348, Aug. 2014. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc7348.html>
- [8] A. Conta and S. Deering, “Generic packet tunneling in ipv6 specification,” Internet Engineering Task Force, RFC 2473, Dec. 1998. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc2473.html>
- [9] P. Savola, “Mtu and fragmentation issues with in-the-network tunneling,” Internet Engineering Task Force, RFC 4459, Apr. 2006. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc4459.html>
- [10] T. Narten, E. Gray, D. Black, L. Fang, L. Kreeger, and M. Napierala, “Problem statement: Overlays for network virtualization,” Internet Engineering Task Force, RFC 7364, Oct. 2014. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc7364.html>