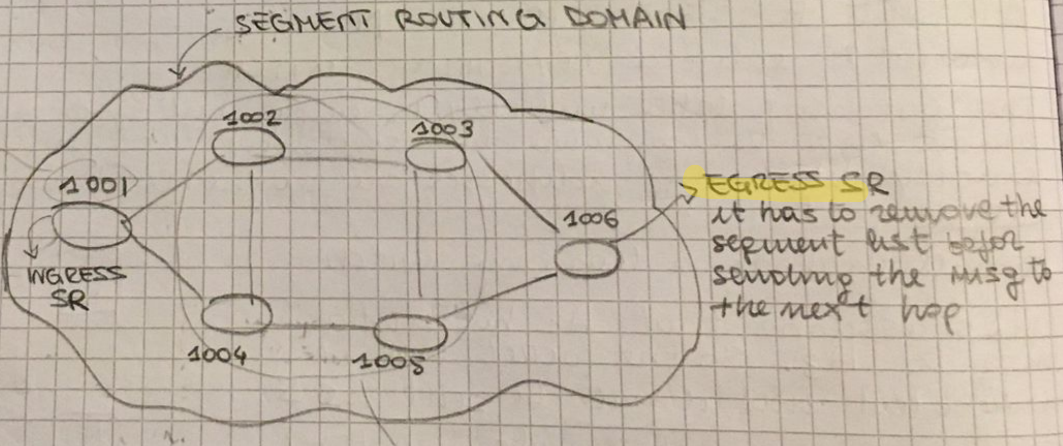
SEGMENT ROUTING

23/12/2020

Is still an ongoing process. Is a way to perform routing.
~~[illegible]~~ We want to realize a source routing supporting architecture.
SOURCE ROUTING: is an environment where the source node is the node where originate the packet.
The source node can decide what is the path to use is by steering it over an ordered list of instructions.
So the SN does not simply select the path, but it can say what ~~actions must be done on a packet~~, functions must be applied on a packet, etc. So it is not simply a routing application but we can decide a set of functions (firewall, intrusion-detection system, deep packet inspection,.....)
The way a source node decides how to apply those functions is to include in the packet itself a list of instructions to be applied which is referred to as SEGMENT LIST.
So a SEGMENT is an instruction while a list of SEGMENTS is a SEGMENT LIST and it is included inside the packet.
This is a difference from OpenFlow switches, in that case all the instructions needed to process the packet are stored in the nodes. (flow rules inside flow tables can be considered as a set of instructions of a programming language), so we can program the switch to act like different boxes. But if we do like this the flow table of the switch will be huge in terms of content.
In this case the flow state is moved from devices to the packet itself. It increases the overhead but it is not a big problem because we have bandwidth to handle it.
On the other hand, update device's memory is a quite expensive problem so is always better to move the flow state on the packet so that we can have lighter device.

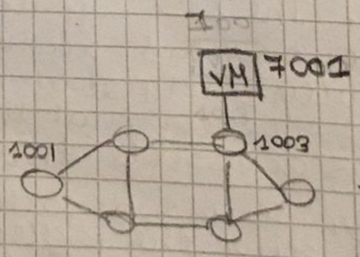
SG TERMINOLOGY

Every node is univoquely identified by a label and every number is a segment (segment identifier or SID)
So the segment list is a list of SIDs and are instructions.

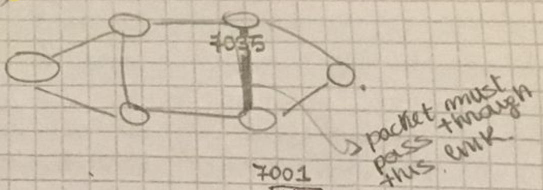

SEGMENT ROUTING DOMAIN
1001
1002
1003
1004
1005
1006
INGRESS SR
EGRESS SR
it has to remove the segment list before sending the msg to the next hop
TRANSIT ROUTERS
(these nodes could not be able to perform second routing related operations

A SID can also be something different, not simply routing applications, not simply forward operations.
For instance I can have access to a virtual machine VM running virtual network function (firewall or other).
So if node 1001 wants specific traffic is steered through this network function, it has to specify 7001

⟹



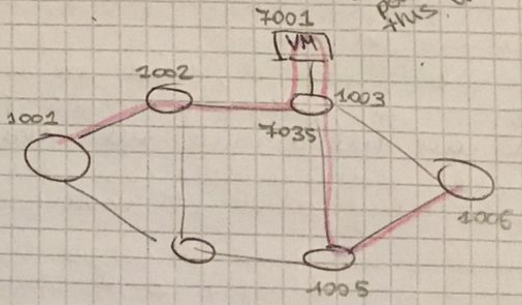
There are also topological SID and (7035)
it forces the packet to go over
the link where it is specified
It is a LOCAL SID or ADJACENCY SID



Now, let's consider that a packet
has to follow Red path,
The segment list will be:

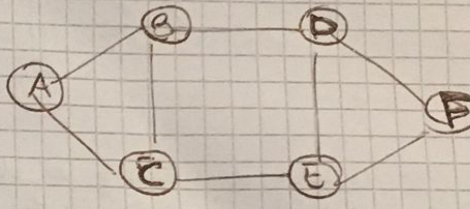
1003
7001
7035
1006
pkt

first goest from 1001 to 1003
then go to VM

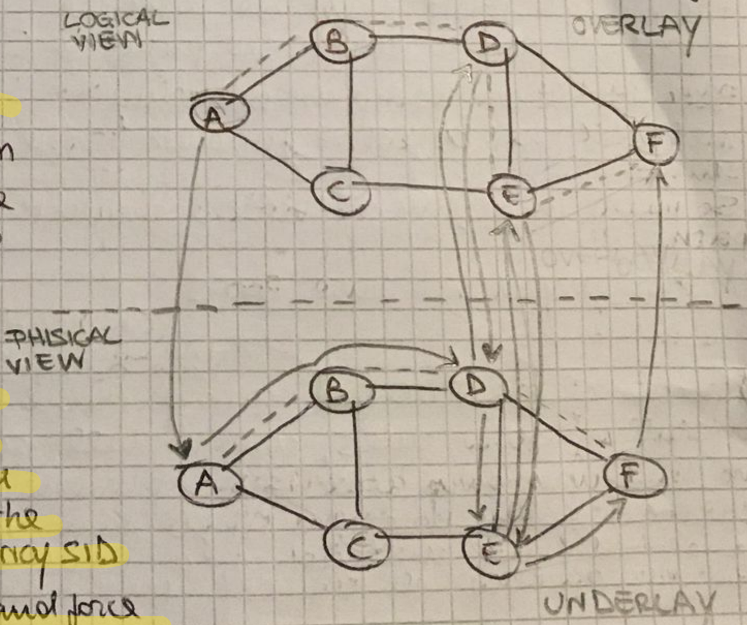


UNDERLAY and OVERLAY NETWORK

Problem: Who tells to routers 1001 and 1002 how to reach 1003?
SEGMENT ~~[illegible]~~ ROUTING is an OVERLAY ARCHITECTURE
So we have a default network and it can
be for instance an IP network.
This network is called UNDERLAY
For IPv4 second routing does not exist,
so for this example let's consider IPv6.
So this is an IPv6 network. Since it is
an IPv6 network, we have routing control plane which is running a
routing protocol like OSPF.
OSPF routers compute the shortest path according to Dijkstra
algorithm.
Every router here is able to send traffic according to OSPF protocol and
on top of that, on the OVERLAY we want to realize a different routing.
How can we do that?



Whereas from A to D is the same
path between over and underlay, then
we can say to A_{overlay} to use path in
the underlay to go from A to D. Once
you have arrived in D, turn back to
the overlay.
This is the first instruction: go to
node D.
Once I am in the overlay I want to
go in E_{overlay}. ~~Stop~~. Assume that D-E
link is very high cost, so it is not in
any shortest path, how can I force the
packet to pass here? Using ADJACENCY SID
So I go back from D in the underlay and force
DE and I go on E_{underlay}.
So now I go from E_{under} to E_{over} and then I ~~go from E to F overlay~~
return back in E_{under} and I find the shortest path to go from E to F.
Then I finally go back to F_{over}



So ~~[illegible]~~ the example before was an example of ~~a~~ how deliver a packet according to a specific segment list.

What about the control plane? SR CONTROL PLANE

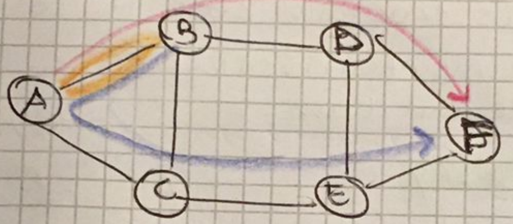
Segment routing is mainly an architecture for data plane, so ~~e~~ it does not really care about control plane, so it does not impose the use of a specific control plane, we can use it either in a distributed way or centralized.

In case of distributed, there are nodes that exchange information using extensions of routing protocols (OSPF and BGP). For example OSPF protocol can tell a router all other router SID.

Every node individually can decide what is the path to use according to a local policy and it individually compute itsown segment list.

Differently from pure IP where the process of calculating the path has to be done accordingly to a common objective; for SEGMENT ROUTING every node can have itsown objective and we will end up with a loop free routing as well. The reason is because of the segment list.

FOR EX



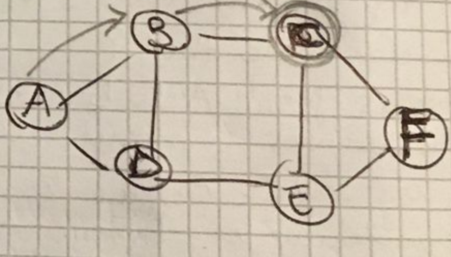
A wants to go to F through the red path, B throug blue path

So between A and B there is a loop.

if we route according to destination address F, in IP we will have a loop, because A goes to B and then B goes to A to reach F

So this shows that in IP we mus use the same objective function.

However, in ~~seg~~ segment routing we don't have this problem and every node can ~~e~~ easily computes itsown segment list according to a local policy that can be possibly different in every single node. And this is because ~~[illegible]~~ from A to F the real destination is not F but C which is the middle point then we find the shortest path to deliver the packet to C



Because we are using an underlay network the objective function is the same.

So in the underlay we use the shortest path.

In this way don't have loops

Differently from SDN in this case we don't have the reactive installation of flow rules because we don't have OpenFlow

In case of centralized control plane like for the SDN, and instead of installing flow rules, can install segment list in the source node and in this way ~~our~~ our goal is to optimize the network performance ~~&~~ instead of the one of a single flow is the best option because we have a global view instead of a local view.

In this approach all segment lists need to be reactively installed ?

In case of an hybrid scenario ~~[illegible]~~ most of the traffic is sent over simple segment list that are calculated locally from nodes. Then some very important traffic flows are optimized and handled by the centralized controller

SR data plane

SR is a data plane technology and defines a new type of data plane with different operations.
So I have to modify all the devices (so I need to buy new routers, define new softwares ~~[illegible]~~ to perform the ~~[illegible]~~ actions).
On the other hand, ~~[illegible]~~ we have to find a set of actions to be performed, packet structure and so on).
I am able to map every function that I propose, over something that already exists. In this case I can implement ~~a new logic~~ the new logic that I want which is segment routing without modifying the ~~[illegible]~~ existing network.
Segment routing by a teoretical point of view is a completely new data plane technology, but from a practical point of view it ~~[illegible]~~ simply reuse the existing data planes. This means that we can make it work in practice without ~~touching~~ touching the existing infrastructure.

We have two different options:
- MPLS (Multi-protocol label switching)
- IPV6

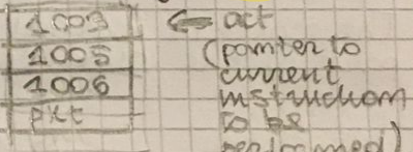
We can't have SR on IPV4 because IPv6 is extensible (we can easily add new features by designing new extension headers)
SR Header (SRH) is a special routing header for IPv6

SR FORWARDING

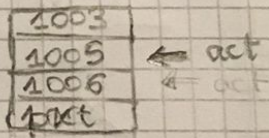
functions used in SR. In normal IP routers nodes have only a single function which is the FORWARD

In case of SR: → push is performed only when we arrive at ~~node~~ node identified by the active segment

- PUSH it is performed by the source node which is the node originating the packet
  This operation consists of the insertion of a segment at the top of the segment list
  (it is a sequence of SIDs)



- NEXT: when we arrive at a node which is identified by the active segment, the operation is the NEXT.
  So we need to modify the position of the pointer.



- CONTINUE: if I go on a node that does not support SR, then I don't need to update the position of the pointer, and I just go on

GLOBAL and LOCAL SEGMENTS

We have two different types of SIDs

- GLOBAL SR IDENTIFIER
- LOCAL SR IDENTIFIER

GLOBAL: is a label that specifies an instruction that is known by everyone (everyone knows that that SID exists and all the nodes know how to perform the instruction related to the SID
(for instance, in the previous example 1001, 1002... where global SIDs)

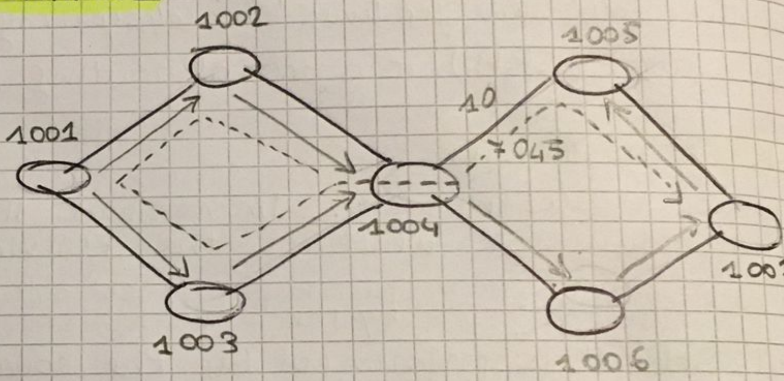
LOCAL: the SID is known by everyone but only one node can perform instructions related to the SID
(for instance, in the previous example 7035 was a local one, so only node 1003 can forwark packets through it)

In order to distinguish between global and local SIDs there are different prefixes, in fact for global start with 1, while local start with 7

~~TYPES OO~~

TYPE OF SEGMENTS : SOME EXAMPLES

We suppose to have this network with a cost =10 on one link to say that it is not part of any shortest path.



~~Suppose we want to~~ [illegible]

Suppose that ⟶ are the shortest paths.

Suppose that we want to realize the - - - - routing

The probelm is that we want to pass throug 7045, but it was not part of the shortest path. So we force it to pass there.

In 1001

1004
7045
1007
pkt

⟶ In 1002 and 1003 I go on. In particular ⟶ in 1004 I can force the packet to go through 7045 ⟶ then I continue in 1005 ⟶ then I arrive in 1007

If in the underlay there is an equal cost multi path, it will distribute fairly over all distinct multi paths