**Concept Notes**

* Segmentation routing is a popular forwarding paradigm for use in MPLS and IPv6 networks
* Multiprotocol Label Switching (MPLS) – is a routing technique that directs data from one node to the next node based on short path labels rather than long network addresses
  + Avoids complex lookups in a routing table
  + Used in telecommunication networks
  + The labels identify virtual links (paths) between distant nodes rather than endpoints
    - Packet-forwarding decisions are made solely on the contents of the label, without the need to examine the packet itself
  + MPLS can encapsulate packets of various network protocols
  + MPLS lies between the link and network layer
* IPv6 – most recent version of the internet protocol
  + IPv6 uses 128-bit addresses, theoretically allowing 2^128 addresses
  + Permits hierarchical address allocation methods that facilitate route aggregation
* BGP Large Community Attribute – Attribute being used in our SR implementation
  + Network operators attach BGP communities to routes to associate particular properties with these routes
    - E.G. route origin location, specific routing policy
  + Each BGP large community value is encoded as a 12-octet quantity
    - Global Administrator: A four-octet namespace identifier
      * This should be the ASN
    - Local Data Part 1: A four-octet Operator-defined value
    - Local Data Part 2: A four-octet Operator-defined value
  + If a range of rotes is aggregated, then the resulting aggregate should have a BGP Large Communities attribute that contains all of the BGP Large Communities attributes from all of the aggregated routes
  + Canonical Representation
    - The canonical representation of BGP Large Communities is three separate unsigned integers in decimal notation in the following order
      * Global Administrator, Local Data 1, Local Data 2
      * 644696:4294967295:2, 64406:0:0
    - No leading zeros

**Segmentation Routing**

* Segment Routing
  + Source Routing
    - The source chooses a path and encodes it in the packet header as an order list of segments
    - The rest of the network executes the encoded instructions
  + Segment – an identifier for any type of instruction
    - Forwarding or service
* Segment Routing – Forwarding Plane
  + MPLS – an ordered list of segments is represented as a stack of labels
    - Segment routing re-uses MPLS data plane without any change
    - Segment represented as MPLS label
    - Applicable to IPv4 and IPv6 address families
  + IPv6 – an ordered list of segments is encoded in a routing extension header
* Global and Local Segments
  + Global Segment
    - Any nod in SR domain understand associated instruction
    - Each node in SR domain installs the associated instruction it its forwarding table
    - MPLS: global label value in Segment Routing Global Block (SRGB)
  + Local Segment
    - Only originating node understands associated instructions
    - MPLS: Locally allocated label
* Global Label indexes
  + Global segments always distributed as label range (SRGB + index)
    - Index must be unique in segment routing domain
  + Best practice – have the same SRGB on all nodes
* IGP Segments
  + Two basic building block distributed by IGP
    - Prefix Segments
    - Adjacency Segments
* IGP Prefix Segment
  + Shortest-path to the IGP prefix
  + Prefix segment is a global segment
  + Label = (SRGB + Index
    - Only advertised as index
  + Distributed by OSPF
* IGP Adjacency Segment
  + Forward on the IGP adjacency
  + Local Segment
  + Advertised as label value
  + Distributed by OSPF
* Combining IGP Segments
  + Steering traffic on any path through the network
  + Path is specified by list of segments in packet header, a stack of labels
  + No path is signaled
  + No per-flow state is created
  + Single protocol: OSPF

**Segmentation Routing in IPv6**

* Segment Routing
  + Source Routing – the topological and service (NFV) path is encoded in the packet header
  + Scalability – the network fabric does not hold any per-flow stated for TE or NFV
  + Simplicity
    - Automation: TILFA
    - Protocol elimination: LDP, RSVP-TE, NSH…
  + End-To-End
    - DC, Metro, WAN
* Network Instruction
  + | Locator | Function(arg) |
    - 128-bit SRv5 SID  
      Locator: routed to the node performing the function
    - Function: any possible function (optional argument)
      * Either local to NPU or app in VM/Container
    - Flexible bit-length selection
* Network Program in the Packet
  + IPv6 Header = | Source Address | | Destination Address |
  + Segment Routing header
    - Next Segment => | Locator 1 | Function 1 |
      * | Locator 2 | Function 2 |
      * | Locator 3 | Function 3 |
  + IPv6 Payload => | TCP, UDP, QUIC… |
* Argument shared between functions
  + Global Argument
    - Metadata TLV
* Group-Based Policy
  + TAG – contained above the segment routing header
* SR Header
  + Slide 19 in SRv6 PowerPoint
* Source Node
  + Source node is SR-capable
  + SR Header (SRH) is created with
    - Segment list in reversed order of the path
      * Segment List [0] is the Last segment
      * Segment List [n-1] is the First Segment
* Non-SR Transit Node
  + Plain IPv6 forwarding solely based on IPv6 DA
  + No SRH inspection or update
* SR Segment Endpoints
  + SR Endpoints: SR-capable nodes whose address is the IP DA
  + SR Endpoints inspect the SRH and do:
    - If segments Left > 0, THEN
      * Decrement segments Left (-1)
      * Update DA with Segment List [Segments Left]
      * Forward according to the new IP DA
    - ELSE (Segments Left = 0)
      * Remove the IP and SR header
      * Process the payload
* Segment Format
  + SRv6 SIDs are 128-bit addresses
    - Locator: most significant bits are used to route the segment to its parent node
    - Function: least significant bits identify that action to be performed on the parent Node
      * Argument (optional): Last bits can be used as a local function argument
  + Flexible bit-length allocation
    - Segment format is local knowledge on the parent node
  + SIDs have to be specifically enabled as such on their parent node
    - A local address is not by default a local SID
    - A local SID does not have to be associated with an interface
* END – Default Endpoint
  + Default endpoint behavior (node segment)
    - Decrement Segments Left, update DA
    - Forward according to new DA
  + Node 2 advertises prefix A2::/64 (A2::/64 is the SID locator)
    - Packets are forwarded to node 2 along the default routes (shortest path)
  + On 2, the default endpoint behavior is associated with ID 0 (0 is the function)
  + The SID corresponding to the default endpoint behavior on node 2 is A2::0
* END.X – Endpoint then Xconnect
  + Endpoint xconnect behavior (Adjacency segment)
    - Decrements segments Left, Update DA
    - Forward on the interface associated with the Xconnect segment
  + Node 3 advertises prefix A3::/64
    - Packets are forwarded to node 3 along the default routes (Shortest path)
  + On 3, the endpoint xconnect behavior for link 1 is associated with ID C1
  + The SID corresponding to endpoint xconnect-1 behavior on node 3 is A3::C1
* END.B6.ENCAPS – Ipv6 Binding Segment (encap)
  + IPv6 binding segment
    - Decrement Segments left update DA
    - Push outer IP and SR headers associated with the binding Segment
    - Forward according to outer header DA (first segment of the new SRH)
  + Node 4 advertises prefix A4::/64
  + The SR encaps policy (SA = A4::, SL = <S1, S2, S3>) is associated with ID 10
  + The corresponding binding SID is A4::10
* END.B6 – Ipv6 Binding Segment (insert)
  + IPv6 binding segment
    - Do not decrement Segments Left
    - Push outer SR header associate with the binding segment
    - Update DA with the first segment of the outer SR header
    - Forward according to outer header DA (first segment of the new SRH)
  + Node 4 advertises prefix A4::/64
  + On 4, the SR insert policy <S1, S2, S3> is associated with ID 20
  + The corresponding binding SID is A4::20
* END.PSP – Penultimate Segment Popping
  + Penultimate Segment Popping (PSP) behavior
    - Decrement Segments Left, Update DA
    - If Segments Left = 0, remove SRH
    - Forward according to new DA
  + Node 5 advertises prefix A5::/64
  + On 5, the Penultimate Segment Popping behavior is associated with ID 1
  + The corresponding SID is A5::1
* END.USP – Ultimate Segment Popping
  + Ultimate Segment Popping (USP) behavior
    - If Segment Left = 0
      * Pop the top SRH
      * Restart the END function processing on the modified packet
        + Decrement Segments Left, Update DA
        + Forward according to the new DA
  + Node 6 advertises prefix A6::/64
  + A6:: is the last segment in the top SRH
* SID Function – Anything
  + SID functions are locally defined on their parent node
    - They can do anything
  + An SR header contains a network program

**Research Paper Draft**

* We are using segment routing and BGP working at the data and control planes respectively to create a novel service specific Interdomain routing framework to define richer interdomain peering policies to enable more fine-grained control on treatment of service specific traffic across the domains
* Achieving a service specific peering using modern paradigms of SDN and segment routing since they are more adoptable and provide more fine grain control
* Redistribution communities and virtual peering’s have also been proposed to enable the ASes to control the flow of their incoming traffic
  + But not good because of certain problems
* Use SDN to tackle the constraints of vanilla BGP which can only forward traffic based on destination IP prefix
  + However, because SDN is data plane, we wat our approach to be defined more broadly in terms of data as well as control plane
* Segment routing offers flexibility and scalability which enables increased variety of cloud-based services supporting string service level agreements
* Using segment routing, network nodes can steer packets over a particular path according to the segments that come up in the packet header
  + A segment represents a function that can be executed at a specific location in the network
  + Segment routing removes the need to implement routing places with per-flow entries since the information of the path that the packet will take is included in the packet header itself
    - Eliminates the need to maintain state
* Using BGP and Segment routing to create a novel service specific inter-domain routing framework
  + BGP at the control plane
  + Segment Routing on the data plane
* For BGP we use Large BGP Community attribute to advertise the service specific routes from the service provider to the subscriber in adjacent or non-adjacent domains allowing network operators to associate particular properties with these routes
  + Route origin location or a specific routing policy action
  + Consists of 3 4-octet values
* The Central Entity of each domain consumes these prefixes and pushes appropriate segment IDs over the packet’s data plane to be routed along the segment routed periphery
* We modified BGP to enable advertisements denoted as Prefix:Tag
* The SDN classifier acts as a gateway for non-segment routing packets to get to the segment routed world
* The Centralized Entity dynamically drives the SDN to bring in the non-SR packets to the SR-world
  + We are pushing segment ID’s to the packet over the data plane
* Centralized Entity in each domain acts as an oracle which has global topological view of the AS as well as knowledge of BGP prefix:tag to segment Routing mappings
* Centralized Entity
  + BGP – this component handles the control plane.
    - The prefixes advertised by service provider in one of the domains reach the service consumer through iBGP or eBGP
    - The BGP prefix along with the community attribute feeds the SDN for appropriate segment IDs to be assigned
  + Service Specific Logic – prefix translation to the appropriate SIDs
    - A prefix along with its community attribute which defines a specific service is mapped to a particular set of SIDs
  + SDN – Maps control plane BGP advertisement to appropriate Segment IDs
    - This occurs by consuming information from Intra-Domain, BGP and the Service Specific Logic components within the entity
  + Intra-Domain – keeps track of all the services routes within the AS
    - It communicates with service specific logic to provide information regarding what intradomain paths are available for a particular service
    - Feeds the SDN with appropriate Intra-Domain SIDs that a specific service should assign for a specific service within the domain
* The centralized Entities can communicate directly through an eBGP like protocol
  + Cleaner but is more complex and difficult to employ
* Or through a standard eBGP protocol through the peering routers
  + Standard vanilla BGP
  + Doesn’t enable the freedom of defining much richer peering policies
* Design Considerations
  + BGP – 12-octet quantity
    - ASX:
      * Refers to the global Administrator namespace identifier
    - Prefix: Treatment Code:
      * Which maps to a particular service and its treatment
      * Operator defined values
    - In our scheme we choose to define the first octet as the namespace identifier of the advertiser and the second and third octets to be the prefix and treatment code for the specific prefix
* Implementation
  + BGP
    - To support advertisements/readability of prefixes along with the usage of BGP large Community attributes
  + Free Range Routing
    - Modified free range routing to support the centralized Entity and their communication
      * Enable to interact with SREXT
    - BGP advertises service specific prefixes with different large community attributes corresponding to a service which is then mapped to data plane segment IDs
  + SREXT Module
    - SREXT is a kernel module providing the basic segment routing functions in addition to more advanced ones
      * If the destination address of the IPv6 packet matches an entry in the local SID table, the associated behavior is applied. Otherwise the packet will go through the kernel’s routing sub-system for normal processing
    - Modified the SREXT module to support additional/inter domain behavior for each packet according to its SID that are specific to our usecases

**SDX – Software Defined Exchange**

* Compelling applications
  + Application-specific peering
  + Inbound traffic engineering
  + Server load balancing
  + Traffic redirection through middleboxes