The Extensible Network

Evolution in Protocol and Data Plane Agility

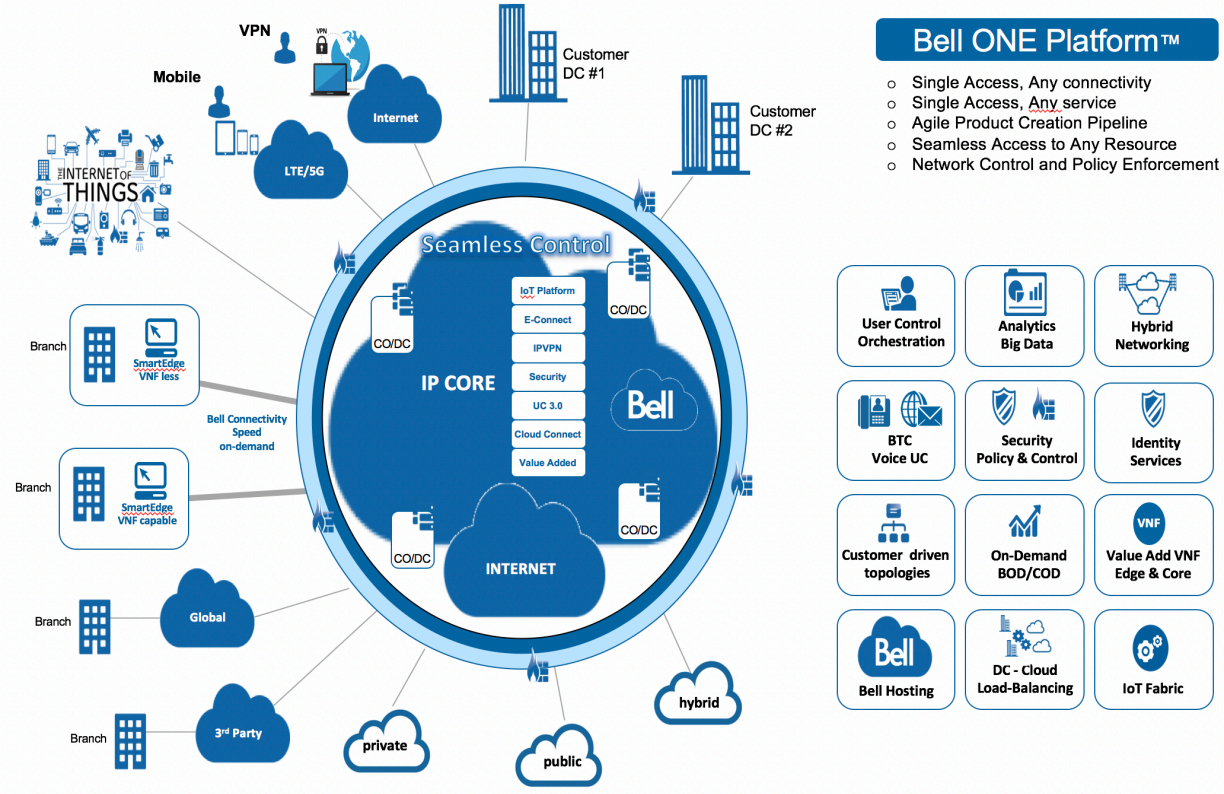
Daniel Bernier, *Bell Canada*; Milad Sharif, *Barefoot Networks*; Clarence Filsfils, *Cisco Systems*

P4 Workshop 2017

A Bit of Context

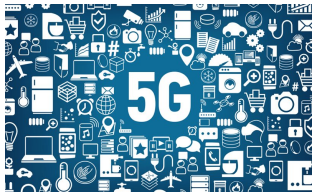
With the **Network 3.0** initiative, Bell Canada is evolving toward heterogeneous, highly interoperable platform that supports an array of relevant business and residential services.

- Large investment in fiber (FTTH, FTTN, etc.) to support future requirements and exponential bandwidth growth.
- Massive network simplification, automation and virtualisation program.
  - Fueled by it's CO/DC transformation (https://www.youtube.com/watch?v=66M8ipFaTeM)
- Disaggregation of connectivity and value-add services from the underlying physical network.

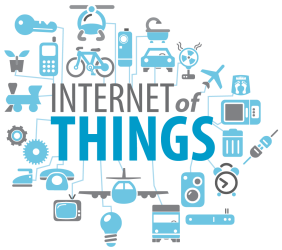


Make the Bell Network Seamless and Personalized

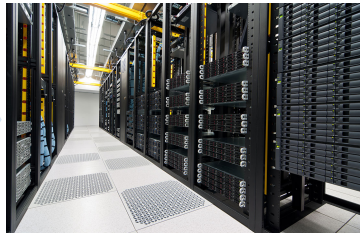
But … "*Winter is Coming*"



5G will tax the network from multiple ends. From low latency and edge proximity to high capacity and flexible orchestration.
The old "throw bandwidth at it" will not suffice.



An estimated 50 Billion devices will be connected by 2020. Interconnecting and securing these at scale cannot use our current network toolkit.



Infrastructure challenges associated to converting 60+ year old central offices to data centers, using commodity hardware has limited the ability for Telco's to leverage proximity to the end users. We are now seeing a return to specialized hardware (FPGAs, GPUs, etc.).

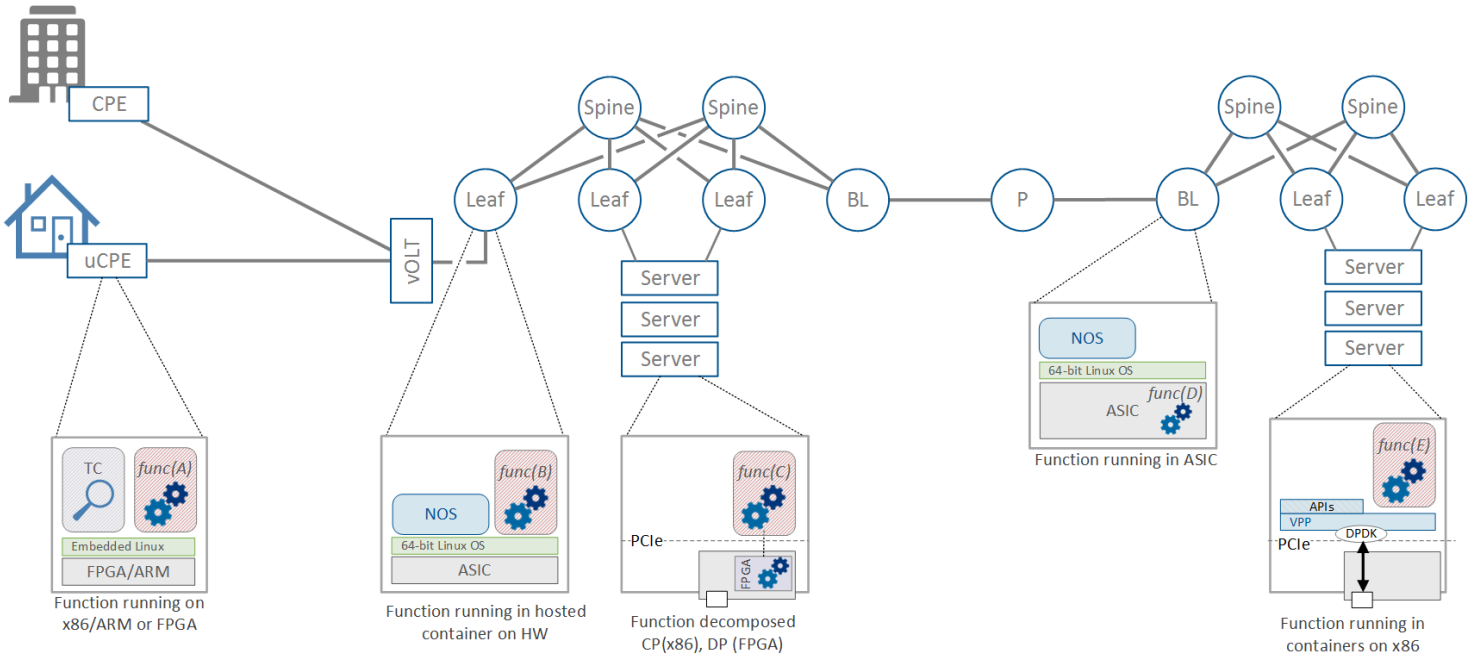
While the industry is still working to virtualize the network efficiently, only few VNFs have moved to micro-services and yet cloud as already moved further (e.g. *FaaS*, *AWS F1*, etc.).

Bell

How Do We Rethink our Network ?

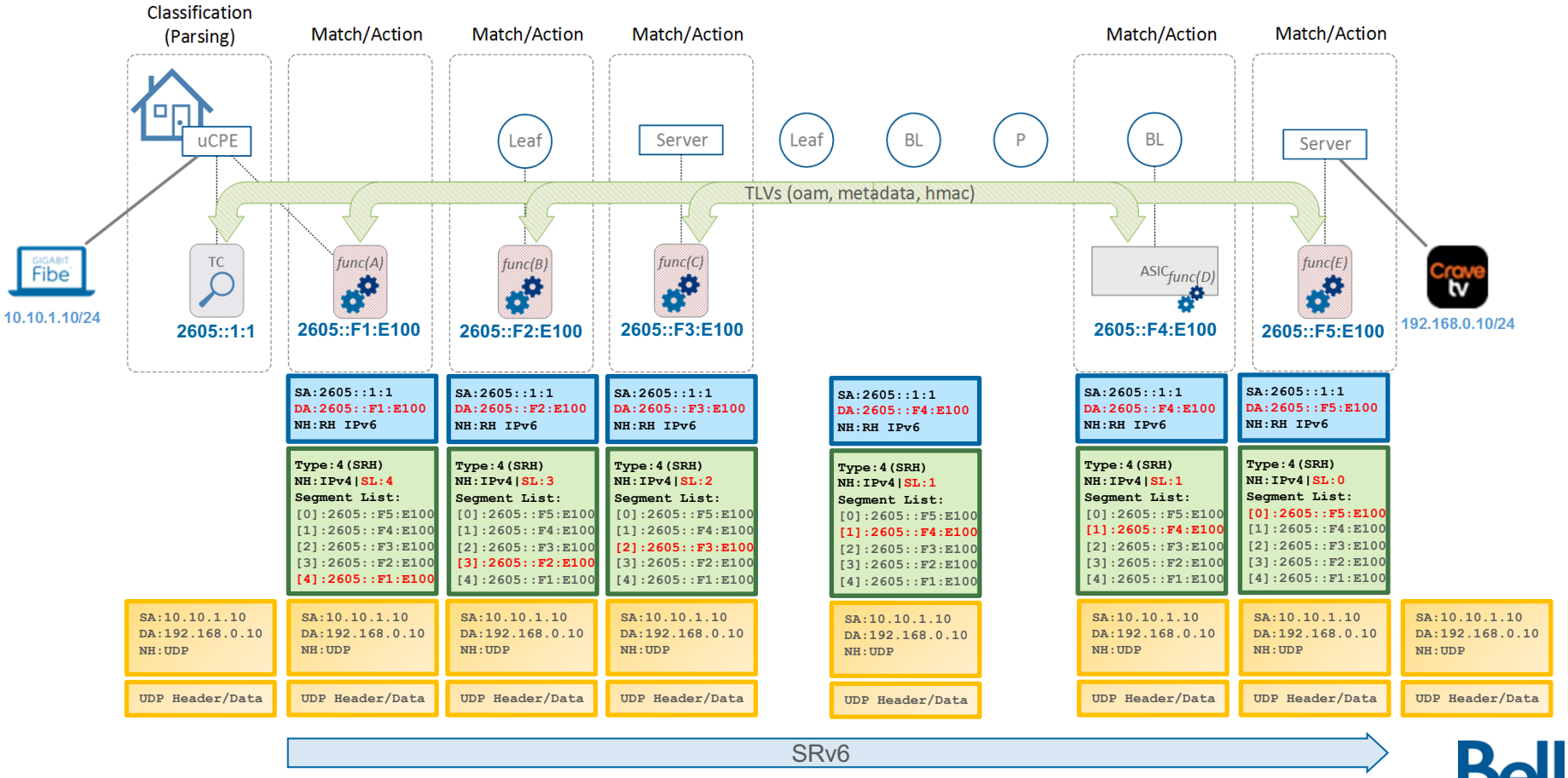
Scaling Things Out

- Make the underlying network stateless
  - Push state to the edges
  - Simplify the protocol soup.
- Distribute functions where they make most sense
  - Functions can be placed anywhere … from network elements to the cloud.
  - That's where a common language for multiple targets comes in handy.
- Distribute function processing
  - 100s of distributed functions will scale better then a few big ones.
- Leverage abstracted function identifiers
  - Make them referenceable and potentially supporting resolution



The "*Network-as-an-ASIC*"

- Traffic classification at the edge of the network → e.g. parsing.
- Simplified *Match/Action* primitive looking at the function Identifier.
- Contextual metadata carried through TLVs
- Programming at all Layers
  - P4 to define the END and TRANSIT behaviors in data plane.
  - SRv6 to define the "*end to end network behavior*"

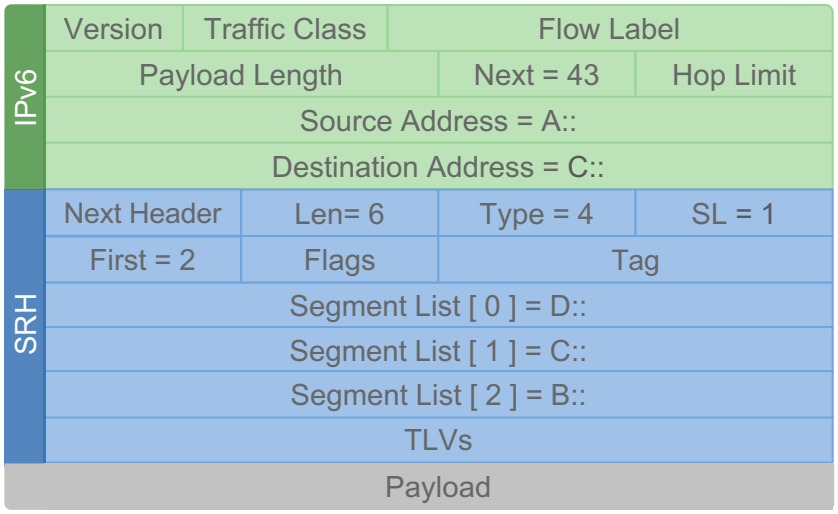


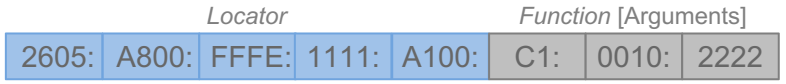
How Can This Be Achieved ?

A Bit of SRv6 Theory

https://tools.ietf.org/html/draft-filsfils-spring-srv6-network-programming-00
https://tools.ietf.org/html/draft-ietf-6man-segment-routing-header-06

- Each segment is an IPv6 Address
- The SID list is encoded in reverse order.
  - **Last segment** in the list as an index value of 0.
  - **First Segment** indexes the first segment in the list.
  - **Segments Left** refers to the active segment index.
- The *Destination Address* of the IPv6 header is copied from the *Active Segment* in the SID list.
- TLVs can be used to convey contextual information.
  - HMAC
  - Metadata
  - OAM values (e.g. INT)
- SRv6 SIDs are 128-bit addresses
  - **Locator**: most significant bits are used to route the segments to its parent node
  - **Function**: least significant bits identify the action to be performed on the parent node
  - **Argument**:[optional] last bits can be used as a local function argument
- Specific SID formatting only needs to be understood by the parent node
- SIDs have to be specifically enabled as such on their parent node
  - A IPv6 address is not necessarily a local SID
  - A local SID is not necessarily bound to an interface.

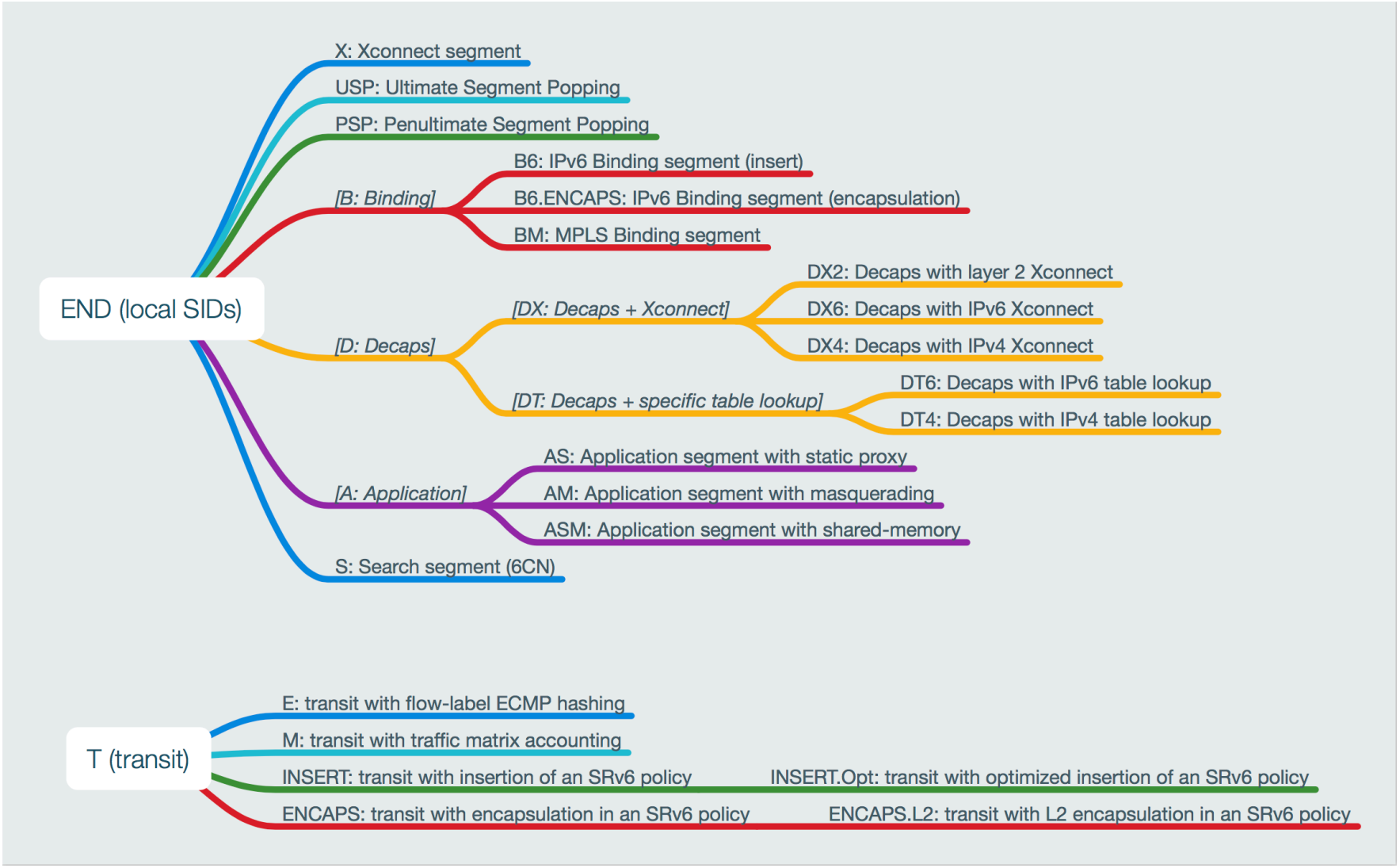




SRv6 SIDs as Universal Identifiers

Bell

Predefined END and TRANSIT Behaviors



Bell

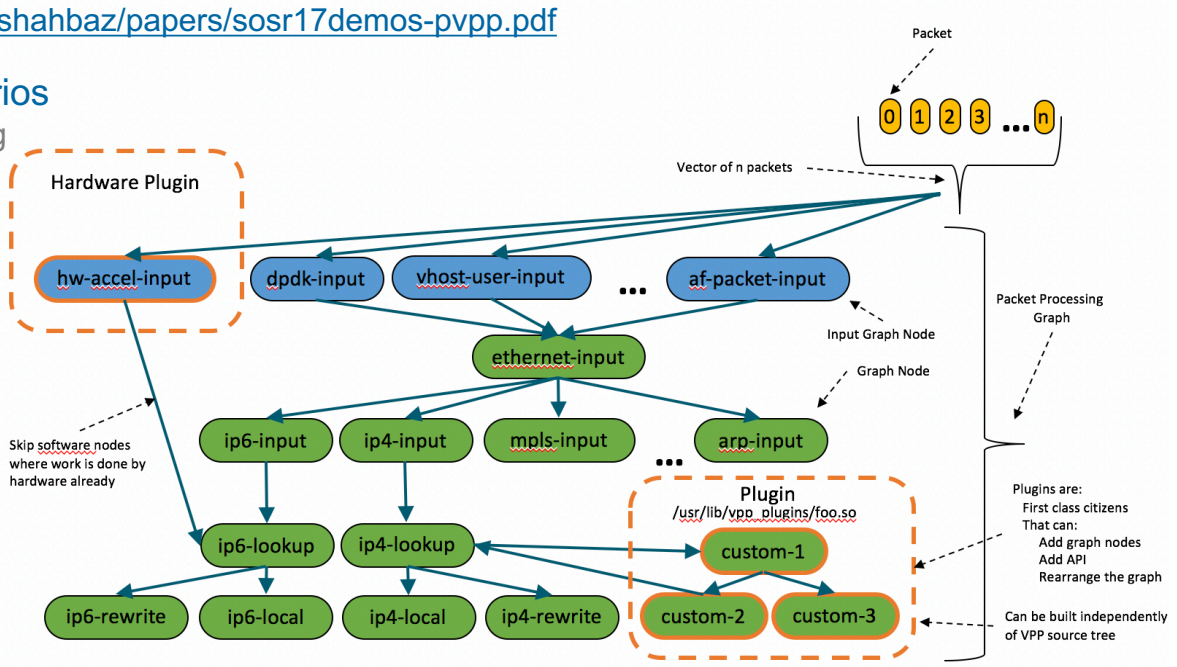
Key Enablers - Software Data Plane (VPP)

- Fully open programmable data plane implementation using directed graphs
- Plugin architecture allows for deviation from core implementation
  - Hardware Offload or Augmentation (e.g. SmartNICs)
  - Capabilities add-on
- SRv6 already Implemented (release 17.04)
  - Sample LocalSID plugin to create new functions

```
set sr encaps source addr A1::
sr policy add bsid B::999:1 next B2:: next D3::2 encap
sr steer l2 GigabitEthernet0/4/0 via sr policy bsid B::999:1
sr localsid address A1::2 behavior end.dx2 GigabitEthernet0/4/0
```

- VPP as a P4 target is a reality

  PVPP: A Programmable Vector Packet Processor (SOSR '17)
  http://www.cs.princeton.edu/~mshahbaz/papers/sosr17demos-pvpp.pdf
- Deployable in multiple scenarios
  - Hypervisor/Host Networking
  - Embedded Systems
  - vNF/cNF Data Plane



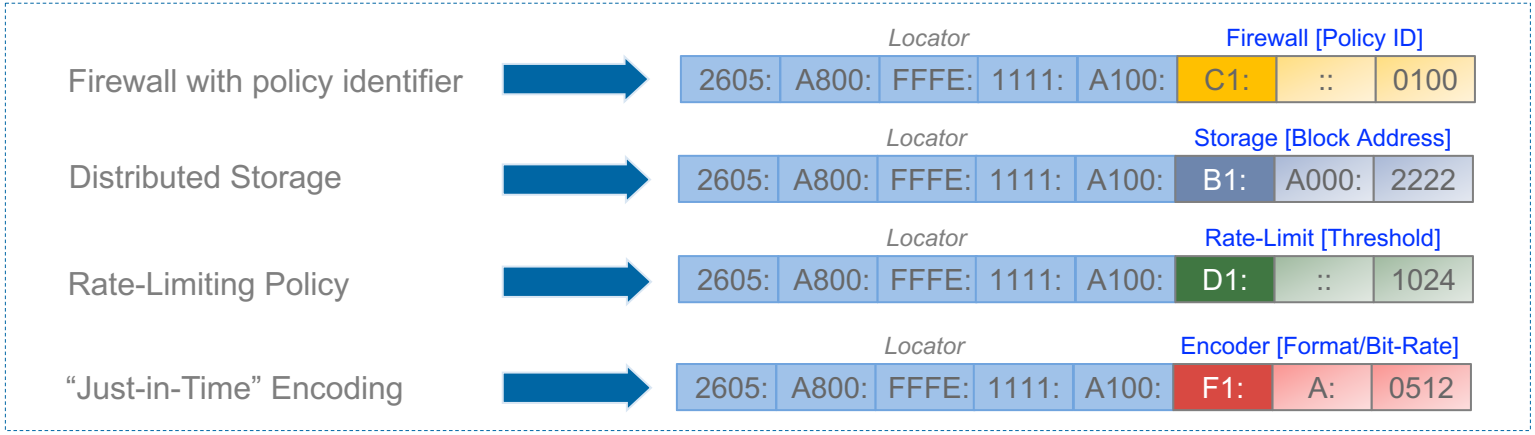
source: https://wiki.fd.io

Bell

Key Enablers - P4 and Programmable Hardware

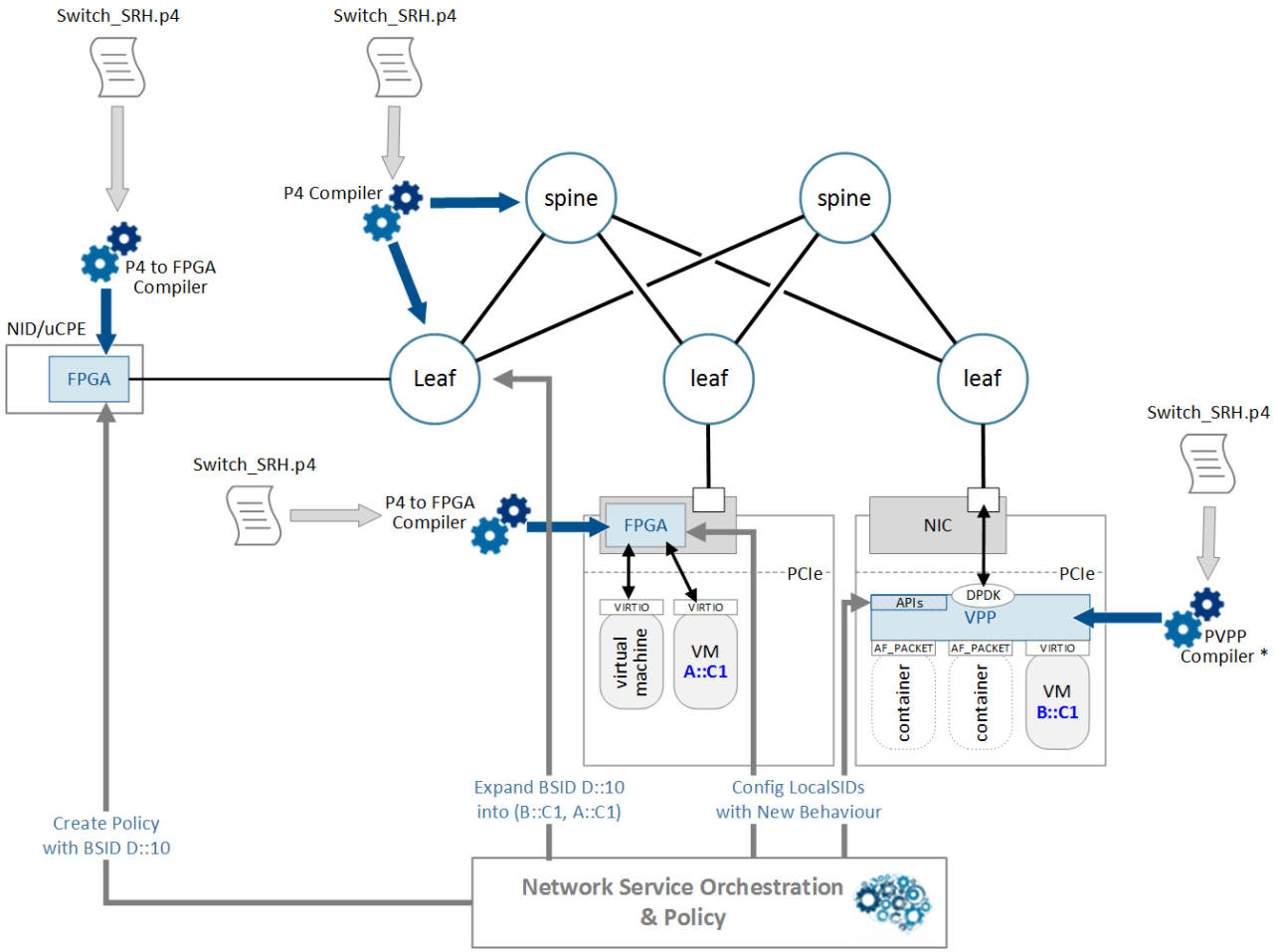
- P4 on SmartNICs (FPGAs, NFPs) can tie END behaviors from the data plane directly to the app.
  - e.g. SR-SPRAY on a NIC
- P4 to FPGA/NPU compilers give access to existing brownfield.
  - Can't replace all the hardware in the field … but some can be reprogrammed.
  - Configuring actions based on an EH might prove simpler than trying to support shiny new encaps.
- Consolidating on P4 helps with usability
  - Keeping developers focused on a single language greatly accelerates innovation.
  - Simpler for end users then support target dependent languages.
  - Builds a larger development community.
- Leveraging *Binding SIDs* can abstract complex policies for resource limited targets.
- Potentially leverage P4 to parse beyond Ethernet (e.g. Optical) ?
  - Opens the door to new possibilities (LocalSID for wavelength?)

Example of potential END behaviors



Bell

Extending the Network With a New Behavior



- Program the new data plane behavior in P4.
- Compile to multiple targets.
- Program the control plane and/or northbound APIs required.
- Configure the new function using the behavior (e.g. attach a LocalSID to the behavior).

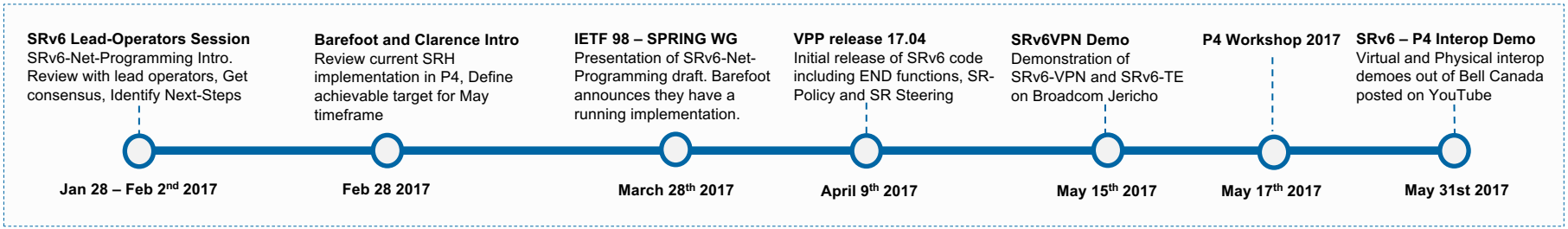
Continuously Extending the Network … With no new Protocol

Bell

SRv6 Implementation in P4

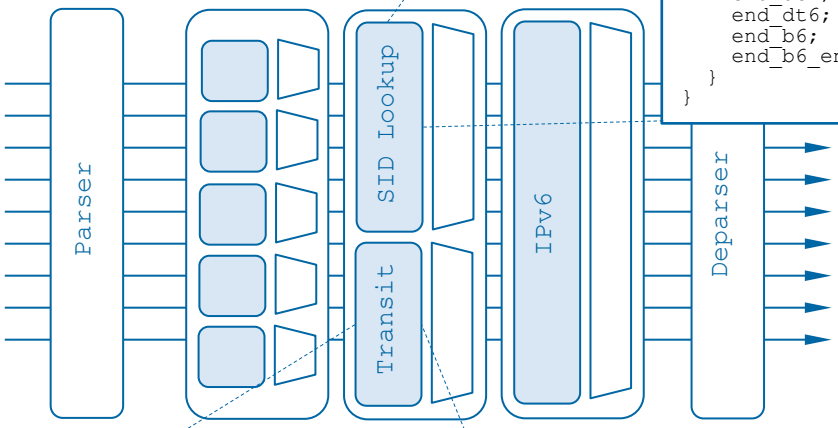
- Idea initiated after the review of the SRv6 Network Programming concept and its surrounding academic work.
  - Lead Operators face to face in January 2017.
- Collaborative work between the SR team at Cisco Systems, Barefoot Networks and Bell Canada.
- Provided a great use case to prove the disruptiveness of PISA.
  - Evaluate behavior of P4 syntax versus hardware taxing encapsulation scheme.
  - Demonstrate rapid innovation open programmable data planes offer.
- Based on SRv6 network programming draft
  - https://tools.ietf.org/html/draft-filsfils-spring-srv6-network-programming-00
  - ~90% of predefined behaviors implemented before draft submission at IETF98
  - **10 segments** in two SRH with L2/L3 forwarding
  - ~400 lines of P4 codes
- Complete virtual and physical interop topologies in Bell Canada labs.
  - Demo will be public in a few week

SRv6 in P4 Development Timeline



SRv6 Implementation in P4

- Concurrent design and implementation.
  - Changes are frequent - 4 revisions of SRH draft in 2017 alone.
  - Can track the latest spec.
- Rapid Prototyping and Testing
  - New SRv6 endpoint functionalities can be added and tested in just a few hours !!
- Zero Effort Silicon Migration
  - Can develop/test with simulation or P4 targets seamlessly.

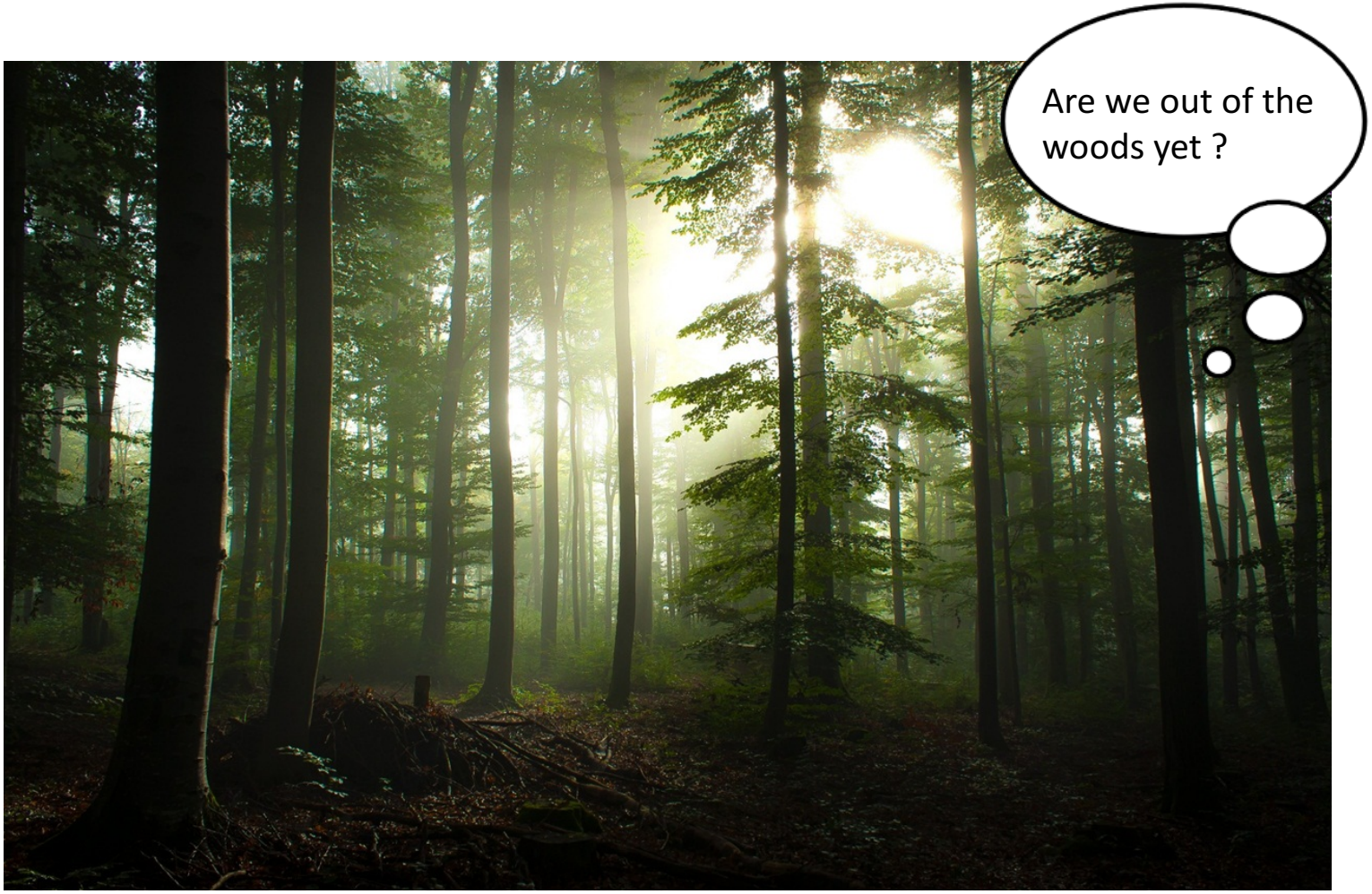



```
table srv6_local_sid {
  reads {
    ipv6.dstAddr : lpm;
    ipv6_srh.valid : ternary;
    ipv6_srh.segLeft : ternary;
    ipv6_srh.nextHdr : ternary;
  }
  actions {
    drop_;
    transit;            /* T, T.INSERT, T.ENCAPS */
    end;                /* END */
    end_x;              /* END.X */
    end_t;              /* END.T */
    end_dx2;            /* END.DX2 */
    end_dx4;            /* END.DX4 */
    end_dx6;            /* END.DX6 */
    end_dt4;            /* END.DT4 */
    end_dt6;            /* END.DT6 */
    end_b6;             /* END.B6 */
    end_b6_encaps;      /* END.B6.ENCAPS */
  }
}
```

```
table srv6_transit {
  reads {
    ipv6.dstAddr : lpm;
  }
  actions {
    t;
    t_insert;
    t_encaps;
  }
}
```


Bell

… it’s not that easy



Still a Lot of Work To Do

Programmable data planes and P4 are key in achieving this *Extensible Network* but there are still a lot of pieces to link together:

- Coding in P4 does not equal to knowing how to program a data plane.
  - Effective data plane programming is an expertise not easily found.
  - Co-development with OEMs will be helpful (code revision, sanity checking, etc.).
  - There will be an industry need for services around P4 code development and multi-target SDE environments.
- A programmable data plane alone does not make a network.
  - We can now rapidly create new SID behaviors to address a requirement or new network function.
  - Leveraging them still needs a control-plane element (e.g. mapping generated APIs to CP, northbound orchestration is essential).
- Lots of standardization efforts at IETF
  - Header insertion is still progressing in the working groups (https://tools.ietf.org/html/draft-ietf-6man-rfc2460bis-11, https://www.ietf.org/id/draft-voyer-6man-extension-header-insertion-00.txt)
  - For interoperability, providers still need standards … but these need to follow the pace of innovation.
  - Programmable data planes give us the ability to move forward iteratively and still reach standardisation (e.g. coding a draft revision in hours instead of months).

Bell

Thank You