NSD – Network Supplied Data

Giles Heron
Principle Engineer, Chief Architect's Office

NSD

- What is NSD?
- What happens today?
- NSD Approach
  - NSD - Message Bus
  - NSD: Provisioning - Service Turn-up (example use case)
  - NSD: Provisioning - Service Change (example use case)
- Challenges and opportunites

What is NSD?

NSD:

- describes the publishing of information by network elements
- focuses on the wealth of untapped data already embedded in the Network today, (Big-Data style)
- helps to centralise the collection and control of the data via simple 'One to Many' publish/subscribe mechanism

- BOF planned for IETF London, March 2-7th 2014

What happens today?

- ‘Traditional tools’ - mostly SNMP or CLI scraping tools which interrogate devices in the network
- In most networks, information is collected by a wide variety of systems –

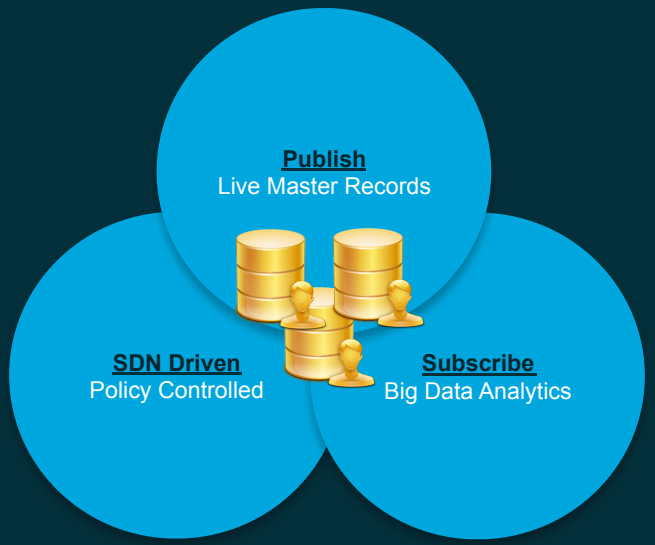
  large scale NMS solutions through to individual systems (many hidden boxes under desks…)
- Security of the information is difficult to maintain

  No central control over the information

  ACLs have to be maintained on devices
- There is no easy way to aggregate information from these disparate systems - Big Data Analytics problem
- There is often no common data format or the format is not web developer friendly (e.g. SNMP uses ASN1 whereas web developers prefer JSON etc.)

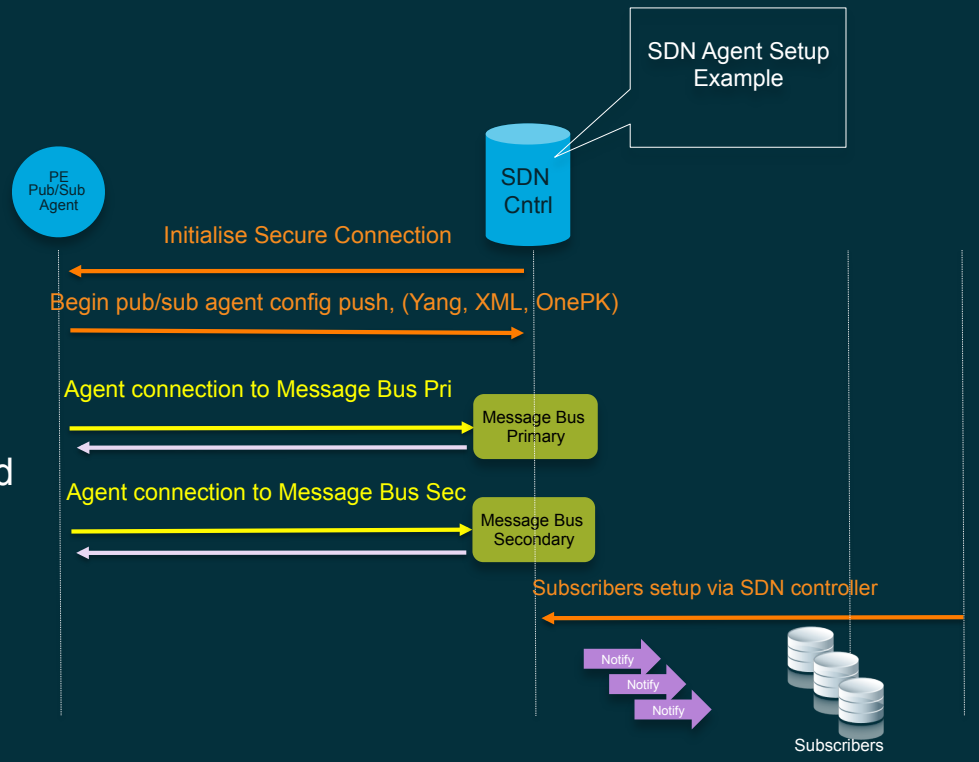
NSD Approach

- NSD is built around the familiar ‘Publish and Subscribe’ concept
- A router/switch publishes information about itself (inventory), events that happen to it, and other information stored in its hardware counters etc
- The information to be published is configured via a central system (SDN controller)
- The information is published using simple lightweight protocols over a secure channel
- The information is delivered to a central store
- Access to the information is controlled centrally via a policy controller
- Analytics tools can access all the centrally stored data
- Uses the network as the **Live Master Record (LMR)**



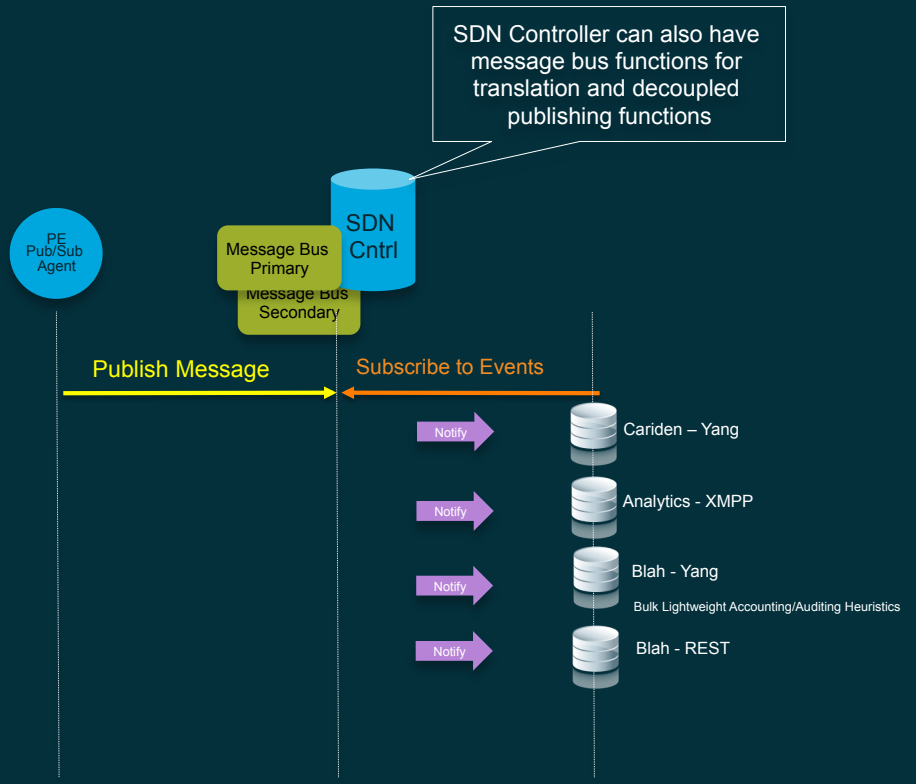
Messaging: Pub/Sub Programming

- The Pub/Sub setup is programmed via the SDN controller.
- Agent key programmatic attributes :
  - Connection/s to pub/sub neighbours (i.e SDN controllers, message bus, other agents).
  - Configuration of events, message types
  - Configuration of security, message transport method
  - Program Message Bus with Notification config



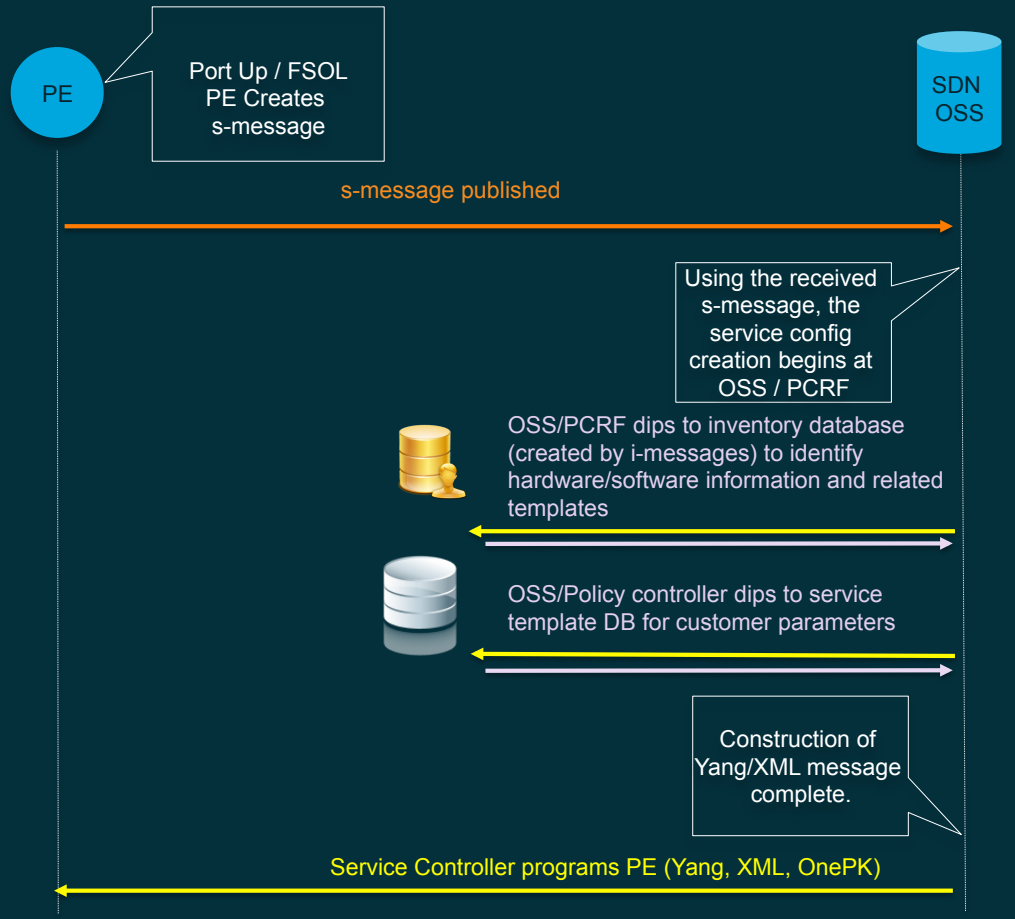
NSD - Message Bus

- The message bus supports both P2MP messaging and also message protocol translation.
  - Message principle of send once, but many possible receivers. (decoupled model)
  - Messages are subscribed to via the message bus, which can be separate or part of the Controller.
  - Different systems are able to have messages delivered / sent in any number of common message formats, as the message bus will translate.
  - Security over the listeners/subscribers is centralised via the Controller.



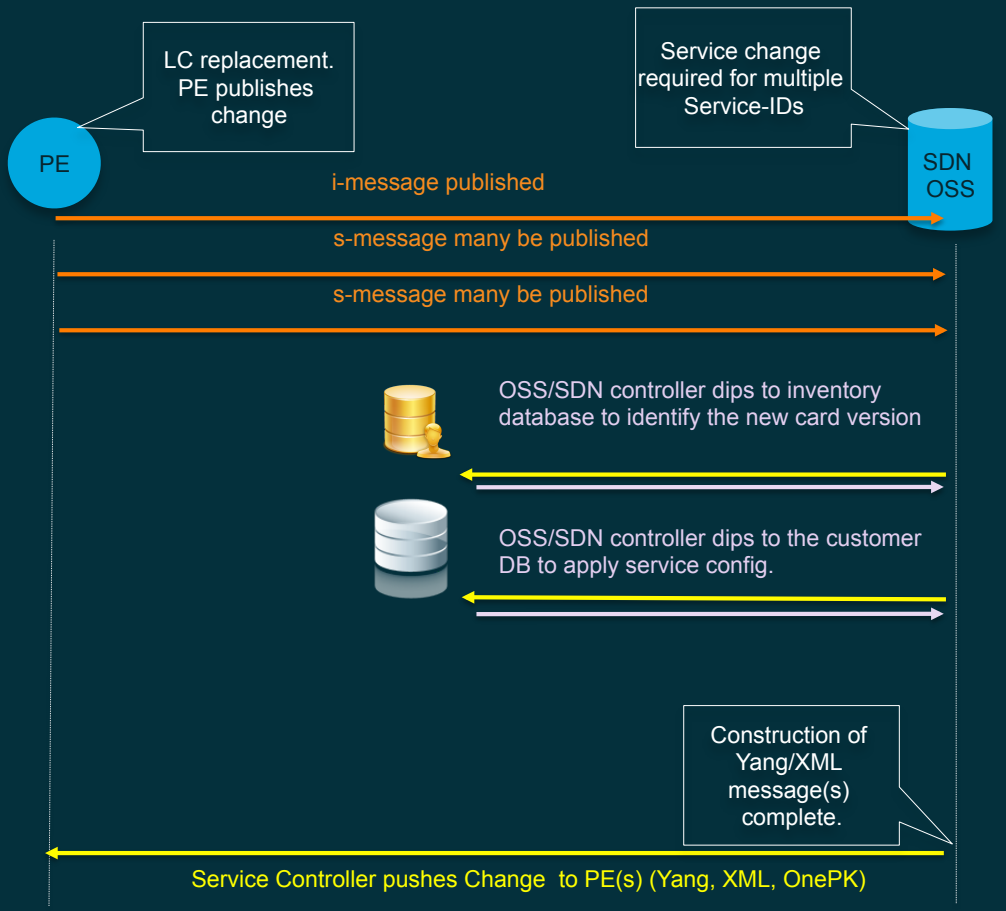
NSD: Provisioning - Service Turn-up (example use case)

- PE publishes a Service ID message (“s-message”) when customer attaches
- Configuration is created based on information in s-message and information previously learned in Inventory messages (i-messages) sent when the PE started up and stored in the LMR DB.
  - Services are built from templates residing at the OSS/PCRF.
  - The Policy Controller constructs the interface/ service config using both the Customer Parameters database and also the Infrastructure database (LMR).



NSD Provisioning - Service Change (example use case)

- Service Change
  - e.g. Linecard upgraded from v1 to v2 in a PE.
  - PE publishes an i-message via the message bus, indicating there has been a linecard change, which updates the network LMR
  - The Controller then identifies the service-IDs related to that card.
  - Pulling together both the templates for the card type and the service-id configs, the OSS/ Controller then publishes the new config to the PE
  - A major issue for SPs is handling both linecard and service-id (customer) changes.



Options for exporting high-volume statistics?

- Common solution is to collect data in the Route-Processor or Linecard CPU and to export it from there
  - RP/LC CPU may be bottleneck when there are millions of counters
  - What is the right format for large volumes of statistics. XML, EXI, something else?
  - Is it better to publish directly or to send a message telling the controller “statistics are ready to be collected”?

- Alternative - stream data directly from the forwarding plane packet engines
  - What capabilities does this require in the packet engine?
  - Do we need to export in a “simple” format and then re-encode to XML, EXI etc. off-box?

Big data – the analytics opportunities…

If we can centralise and normalise the data we can open the door to cross platform and cross function analytics

- Any data can be consumed by any (allowed) application
- Message content is described via XML and/or JSON – therefore web developer friendly
  - Developer doesn't have to know a network transport encoding to get at the event
- Apps can 'mash up' information from completely different layers of the network
  - For example correlate transport network issues to IP reachability problems

Big data – the analytics opportunities…

- Apps can import information completely unrelated to the network
  - For example, correlate road maintenance database with network outages – JCB through the fibre duct…
- Apps can correlate inventory data to event and service data
  - For example correlate intermittent customer disconnects to particular piece of hardware based on serial number
- Real-time 'triggers' can be created by an App
  - When your phone's location changes from work wifi to home wifi, turn on the lights

Real-World Examples using NSD

Service Provider wants to give the same QoS to a device whether it is at home or at a coffee shop using their WiFi service

- Publish event from the home g/w or AP (or upstream router) when the device attaches
- Application running on SDN controller matches device ID (e.g. MAC address) to subscriber QoS profile and provisions home g/w or AP
- SDN controller gathers real time location info to deliver appropriate higher-layer services
- Optionally export usage data to count against volume cap etc

- In essence is we can move to an event / notification driven model, which is not based on the polling and heavy-weight systems of today then we can enable a huge variety of new services

Thank you.