Intent As The Common Interface to Network Resources

Dave Lenrow

HP

Intent Based Network Summit 2015

Palo Alto, CA, USA

Event Background

- Thanks to Keith Burns, Cisco – “Co-inventor” of the intent summit
- Thanks to Mat Mathews, Plexxi – Asked me to find like minded individuals and start a movement almost 3 years ago. The job’s not finished.
- Urgency to start this conversation across the diverse projects looking at Intent.
- The longer we head in different directions, the harder it will be to find common ground.
- Goal of this event is to get folks asking “How can my innovations and use cases be supported by a community supported interface?”
- Group conversation more important than perfect venue or slideware.

Intent Defined and Positioned

- Intent: “what”, not “how”
- Intent as the “universal language”
- Intent is invariant
- Intent is portable
- Intent is compose-able
- Intent is scale-able
- Intent Brings Context

Intent versus Prescription

Intent	**Prescription**
• What I want, not how to do it	• How to do it (Commands, rules, settings)
• Portable, independent of protocol, vendor, media, etc.	• Non-portable, dependent on protocol, vendor, media, etc.
• “I want my headache to stop”	• “Give me two aspirin”
• “Bob is allowed to access the internet”	• “Send packets matching this 5-tuple out port 11”
• “Please cut my lawn”	• “Take mower out of truck, fill gas and oil, pull starter cord, push onto lawn, …”

Common Intent Interface

- If we can get a single interface across multiple vendors, devices, and protocols, we get a network effect:
  - More people adopt common interface in order to have opportunities to participate in the growing ecosystem around it
  - Developers attract more users, users attract more developers, leading to exponential growth of the ecosystem
- It is far more important to have a diverse ecosystem of players sharing a common interface than it is to pick the perfect interface definition
- Winning is defined by success as an industry segment, not the glory of a single individual “inventor” or “vendor”
- By allowing operators to easily compare and contrast solution benefits, we make the “pie” bigger for everyone.

Intent Is Invariant

- Intent doesn’t change as a result of:
  - Link, switch, router, server, storage fault
  - Changing network providers, equipment manufacturers, protocols, devices

- Intent can change dynamically, but a given expression of intent is invariant.
- If it changes as a result of changes in the infrastructure it isn’t intent
- There’s plenty of complicated stuff in the infrastructure that is not invariant and it needs to be configured. Let’s just make sure that what we expose to the high level consumer of the network services is completely abstracted for simplification.

Intent is Portable

- Because intent is invariant with respect to implementation choices, it is portable across implementation choices and changes.
- Reduces friction of changing infrastructure
- Enables the buy-side “holy grail”; Apples vs. Apples comparison of heterogeneous solutions without multiple investments in infrastructure-specific integration.
- Portability to support easy comparison of competing network solutions is a primary goal of the Intent project.

Intent is compose-able

- Disparate, independently developed SDN services can be combined arbitrarily within an SDN domain.
- Intent becomes the common “language” spoken when defining resource requirements
- Extensible Intent rendering system includes logic that understands how to translate intent into resource allocation, detecting and resolving conflicts
- Intent-only SDN NBI avoids the split-brain/multiple-writers problem that otherwise requires profound distributed systems solutions.

Intent Is Scalable (Shard-able)

- Intent is unchanged whether a single SDN controller supports one million ports or a thousand controllers each control a thousand ports.
- Some knowledge of instances (e.g. end point registry) must be shared using (eventually consistent?) shared/global state, but the intent doesn’t need real-time synchronization and can be converted (rendered) locally, supporting linear scale-out.
- Smart top-level controller can “shard” Intent based on state of end-to-end paths. Global intent can be pushed to all domains for redundancy and autonomy.
- Massive Intent domains can be built, while supporting small failure and maintenance domains.

Intent Brings Context

- Key to detecting and resolving conflicts is to be able to examine the “what I need” description, rather than the “how to do it” description.
- Intent allows latitude to satisfy “what I need” in the most efficient manner possible.
- Flow rules and protocol configurations don’t have any context and thus make it impossible to correctly identify or correct conflicts

Resolving conflicts – Service Chaining Example

Prescription

- New rule
  - Match: 5-tuple-A, Action: forward on port 12 (forward towards VF1).
- Existing rule
  - Match: 5-tuple-A, Action: forward on port 11 (forwards toward Internet) .
- Analysis: “There is a conflict”
- Resolution: Not possible, no context

Intent

- New rule
  - When members of sales group access Internet send traffic through VF1, VF2 & VF3
- Existing rule:
  - Members of the sales group are allowed to access the Internet
- Analysis: “There is no conflict”
- Resolution: Render Intent into rules

Intent Only – A proposed solution

- Computer Science 101 says a single writer of device rules needs to arbitrate between needs of the disparate services
- Needs of apps need to be expressed to arbitrator as intent not prescription
- Individual apps cannot be allowed to talk directly to flow rules manager or devices
- Intent must become the only interface to the SDN black box
- Assertion: For any prescription you develop it is possible to create an expression of the intent that was the original requirement and support this by extending the rendering system.

Scope of Intent

- Intent is completely abstract. It does not include references to specific instances of infrastructure devices or state.
- Border interfaces and explicit underlay control cannot be prescribed via intent based interface (Allows for separate infrastructure policy mapped to intent abstractions)
- Not all Policy based infrastructure or networking is Intent Based
- All Intent based networking could be said to be policy based
- Intent becomes the only interface to the pool of infrastructure (network) resources under control. (Again, infra policy, is another input to intent rendering and can thus determine what pool of virtual resources looks like when rendered).

Hard Problems Left To Solve

- How do we make it easy to extend the intent language?
- How do we allow developers to extend intent engine to support new types of intent without having to understand all of the other intent types and use cases?
- Can we detect and resolve all conflicts above the renderer so that rendering logic can ignore lots of potentially complex error checking and such? Or do we need to resolve conflicts at multiple layers in the stack?
- Where do we draw the boundaries between the various modules to enable maximum convenience for developers and maximum interoperability between implementations?