JOHNS HOPKINS
APPLIED PHYSICS LABORATORY

Bundle Protocol Endpoint ID Patterns

IETF 120 DTN WG

Brian Sipos
JHU/APL

Background

- Use cases on the following slide motivate the need for a mechanism to define a set of EIDs in a structured way
  - Goal is to ensure the writer and the user have the *same interpretation*
- Simple globs or regular expressions could be used, but these are not ideal
  - Purely text-based
  - Do not take advantage of the structure for DTN or IPN schemes
  - Do not handle numeric intervals for IPN scheme
  - Do not have an efficient binary encoding
- Pattern matching syntax has a “network effect”
  - The more tools that use a common syntax the more value it has
  - If established, new tools do not need to reinvent a robust mechanism
  - Lessens the possibility of security vulnerabilities from misconfiguration “is this parameter an EID or some glob expression?”
- This proposal is compatible with IPN Scheme update draft-ietf-dtn-ipn-update

Use Cases

- Security identities
  - Allow a certificate holder to be authorized to sign for `dtn://node/**` or for `ipn:3.*.*` or even `ipn:3.*.0`
  - The same way as wildcard certificates, it is a CA obligation to ensure endpoint ownership of all matching EIDs
- Routed blocks and authorization
  - EID Patterns are meant for a more structured situation than “huge list of EIDs”
  - The same purpose as IP CIDR notation *e.g.* `192.168.30.0/24`
- BP Agent configuration / policy
  - Allow BPA configuration to use consistent pattern syntax
  - Allow node `ipn:3.5.0` to sign bundles from `ipn:3.*.*`
  - Provide the same kind of ubiquity as CIDR does for IP configuration
  - Avoids policy engines with over-restrictive or limited expressive syntax
- Colloquial use
  - Have an understandable way to convey technical comments like:
    I’m having trouble sending to ipn:3..**
    Please allocate your services within ipn:.*.[5-10]*

Current State

- Draft in draft-sipos-dtn-eid-pattern-02 with pending issues in https://github.com/BrianSipos/dtn-eid-pattern/issues
- Any-scheme pattern `*:**`
- Any-SSP pattern `ipn:**` or `2:**`
- IPN Scheme Patterns
  - Allow a match-all syntax `ipn:**`
  - Separate the EID into single-integer parts, each part can be one of:
    - Exact-match value (compared as integer)
    - Match-all one-part wildcard
    - Range expression (set of discrete intervals)
  - Compressed CBOR encoding using integers
  - Simple set logic (“Pattern A contains B” or “Pattern A overlaps with B”)
- DTN Scheme Patterns
  - Allow a match-all syntax `dtn:**`
  - Separate the EID into node-name and service-path segment
  - Each part can be one of:
    - Exact-match literal
    - Match-all one-part wildcard
    - Match-any-parts wildcard
    - Regular expression, percent-encoded
  - Complex or unavailable set logic (related to regular expressions)

Last Changes

- Added a section to explain “Goals” of this work
- Made the any-scheme pattern text consistent with the rest
- Added an any-SSP pattern to allow configuration for unhandled but known schemes (“my BPA doesn’t know but I know”)
- Added a use for EID Patterns within PKIX Name Constraints extension to delegate specific naming authority to specific CAs

Feedback on Current Proposals

- Recommendation to switch to using "An Interoperable Regexp Format" from RFC 9485
- Recommendation to disallow empty regexp in DTN-scheme pattern
- DTN-scheme “authority” structure is still undefined and unknown
  - This causes design and usability problems because we don’t have a “typical” or “expected” case of what is being matched and how it is organized

Examples of EID Patterns

- Singleton pattern:
  `dtn://node-name/serv ipn:3.10.5`
- All services on a node
  `dtn://node-name/** ipn:3.10.*`
- One service on any node
  `dtn://**/serv/name ipn:*.*.5`
- Complex wildcard patterns
  `dtn://**/prefix/* ipn:3.*.5 ipn:3.*.*`
- Expressions and ranges
  `dtn://[prefix.*]/serv ipn:3.[5-10,100-110].5`
- Mixed patterns
  `dtn://[node%5BA-Z%5D]/** ipn:3.[10,12,14].*`
- Multiple combined patterns with pipe separator
  `ipn:3.[10,12,14].*|ipn:[4-5].*.*`
- Match-all pattern:
  `*:**`

Considerations

- An EID Pattern *is not* an EID, they cannot be used interchangeably
  - This is a security risk *a la* the wildcard DNS names in early PKIX certificates
  - The syntax has been designed that a range (IPN) or expression (DTN) is specifically *not* a valid EID value per the ABNF syntax
- An EID Pattern is a superset of EIDs
  - It is a design goal that an EID *is* a singleton-matching pattern for itself
- Patterns are conceptually simple but can be complex in practice
  - A common specification can allow shared-use implementations
- IPN pattern special considerations
  - IPN scheme now has three logical parts, IPN patterns always have exactly three components

Next Steps

- Incorporate feedback where possible
- Proposed to separate DTN-scheme EID Pattern to separate doc to not hold up the main capability
- Trial or example implementations
  - Existing BPAs that want to try out this syntax?
  - Potential hackathon topic?