ICN and DTN
NetInf over BP using BPQ

Elwyn Davies

Folly Consulting Ltd/Trinity College Dublin

elwynd@folly.org.uk or davieseb@scss.tcd.ie

01/08/13

Version 0.3

NetInf ICN Protocol

- Short for Net*work of* Inf*formation* Protocol
- Developed during SAIL FP7 Project
- Documented in draft-kutscher-icnrg-netinf-proto-01
- Uses convergence layer (CL) architecture
  - Similar to but subtly different from BP
  - No fragmentation
- Draft defines CLs for IP networks (HTTP/TCP, UDP)
- This work extends NetInf with
  - a CL over BP
  - a gateway between HTTP/TCP and BP.

Basics of NetInf

- Works on Named Data Objects (NDOs)
- Named typically with ni (Network of Inf) URIs
  - See RFC 6920 (Naming Things with Hashes)
- Three pairs of messages
  - GET/GET_RESP
  - SEARCH/SEARCH_RESP
  - PUBLISH/PUBLISH_RESP
- Messages typically carry..
  - NDO name or serach string
  - Whole of an NDO's content or none of it
  - Message ID to link message and response

NetInf over BP

- Uses BPQ block to carry
  - ni name or search string
  - Identification of type of NetInf message
  - Message ID
- Source and Destination Ids as in usual BP
  - Discussion about addressing & routing later!
- Metadata carried as JSON string encoded object
  - Using Metadata block with new ontology
- Some messages use another new Metadata block type to indicate they are not carrying an NDO object

BPQ Block Contents - Updated

- **BPQ-kind**: 4 values defined: query, response, response – do not cache fragments, publish
- **Matching rule type**: tells routers how to match BPQ-value from query with BPQ of a response
  - Only rule type 0x00 (exact match) defined
  - Added keyword search rule for searches
  - rule MUST be the same in query & response.
- **Original source EID and creation info in responses**
- **BPQ-value**: String to be matched
- **Msg-ID**: Added – to carry NetInf message ID
- **Fragment List**: if only returning parts of response

NetInf CL using DTN/BPQ

- Basic idea: encode ni name for bundle (payload) in BPQ-value
- Send GET and SEARCH messages with BPQ-kind set to 'query'
- Return responses with BPQ-kind 'response'
- Any bundle with a BPQ block with BPQ-kind of 'response' passing through a node will be recorded in the node's BPQ cache for as long as the bundle remains unexpired and resources are available

Sending GET Messages

- NetInf GET messages sent in a bundle with an empty payload and a BPQ extension block
- The BPQ-kind used is 'query'
- BPQ block contains a ni name for an NDO in the BPQ-value
- Uses the exact match rule to find cached bundles with the same ni name in their BPQ block.

Making responses to GET

- When a bundle with a BPQ-kind of query arrives in a node
  - Apply matching rule against bundles in cache
- If any match, generate response bundle
  - May contain complete payload or alternatively whatever fragments of bundle are in cache
- Send bundle to originator (source) of query with
  - Source of bundle set to node doing match
  - Original source info for response in the BPQ block (EID and creation timestamp/seq #)
- Forward the query only if the response was not the complete bundle

Caching/Updating a Response

- When a bundle with a BPQ block with BPQ-kind 'response' arrives at a node:
  - Check if all or part of the bundle are in the local cache
  - If so, merge the incoming and cached parts
  - Update the cache with the merged bundle or the incoming response if not already in cache
- Send on the response with the updated entry from the local cache

Delivery of Responses

- On arrival at destination pass the bundle including BPQ block to relevant registered application
  - Note that the sending application would (generally) have created the query with a BPQ block so should be expecting a BPQ block in response
- Also cache response as at any other node

Routing of Requests
(under development)

- Destination for requests will usually be a generic NetInf specific EID
  - For GET dtn://get.netinf, etc.
- Suggested implemnation as multipoint EID
- Request will be sent to all nodes advertising the appropriate generic EID - multicast!
- In local distribution, policy will determine how many hops requests are forwarded
- May also be able to use specific locators (EIDs) if known for NDO wanted - unicast!

NetInf Auxiliary/Metadata

- NetInf messages contain various auxiliary and/or metadata items including
  - locators
  - ext(ension) items
  - search keywords
- These items will be JSON encoded in a string and communicated in a DTN Metadata block
  - See RFC 6258
  - The metadata type code used is 192 (private) initially – permanent code to be obtained later

Practical Fixes to DTN2

- Fixed API handling of Extension and Metadata blocks – previously didn't match spec
- Added JSON handler to application library
- 'Completed' Python and other scripting interface shims
  - Previously didn't handle Extension or Metdata Blocks at all (nasty surprise!)
- Modified various applications to handle metadata blocks and extension blocks correctly
- Added metadata to ICN applications

NetInf BP/HTTP Gateway

- NetInf code available in Sourceforge
  - Https://sourceforge.net/projects/netinf
- Primarily implementations of HTTP CL in various languages
- BP/HTTP gateway in Python code

sourceforge.net/projects/netinf

NetInf BP to Filing System

- Implemented a FUSE filing system
  - Maps NetInf Bundles to files named with ni name
  - Publish file by writing into filing system
  - Later allow GET by reading non-existent file
- In NetInf code available in Sourceforge
  - Https://sourceforge.net/projects/netinf

sourceforge.net/projects/netinf

Status as of 2013/08/01

- Complete BPQ processing in DTN2
- Get/Search/Publish functionality in place
- Lucene search engine functional
- Python HTTP library complete
- Gateway mostly implemented
- Metadata in DTN2 implemented
- Routing still needs to be addressed

Thank You

Questions?

BACKUP

Searches

- For search requests the BPQ query message may contain a string to be fed to a search mechanism; this is carried in the BPQ-value
- The search string should also be returned in the metadata
- TCD have implemented a Lucene search mechanism together with an external view of the cached response bundles that can be invoked at any node that the SEARCH request reaches

Responses to Search

- If any bundles containing NDOs are found to match the search parameters, a list of the relevant ni names is sent back as the payload of a response bundle
- The BPQ-value contains the search parameters that resulted in the list.
- These responses may also be cached (determined by node policy).

Publish

- The BPQ kind 'publish' is an addition to the system not described in the current draft.
- Any node that wishes to publish an NDO generates the ni name and packages the data into a bundle with the name in the BPQ-value of BPQ block.
- The bundle is sent to dtn://publish.netinf and will be delivered to local nodes via 'multicast'.
- Nodes receiving this bundle treat it in the same way as a response bundle and cache the NDO

NetInf Device using DTN/BPQ

- TCD are developing a NetInf device that uses the DTN/BPQ CL (convergence layer)
- The overall architecture of the device is shown in the next slide
- Able to access bundle store via a FUSE user space filing system
  - Will be publishing (Python) code for this shortly
- Apply Lucene search capability to bundles in the store

NetInf DTN/BPQ to HTTP Gateway

- A gateway between domains using the NetInf HTTP convergence layer and the DTN/BPQ convergence layer will be implemented using the Python NetInf library components
- The Python niserver will be extended to convert incoming messages from HTTP to DTN/BPQ and send them into the DTN domain.
- The HTTP client will be used for messages originated in the DTN domain