Network Coding Architecture
- Use cases, protocols and building blocks -

Victor Firoiu, Vincent Roca, Brian Adamson, Marie-Jose Monpetit
Draft 2014-11-11
Overview

• Goal: Present ideas for Network Coding Architecture
• Ideally, this architecture would accommodate all possible use cases – not practical
• Start with several use cases with potential for practical applications, such as existing implementations
• Foster innovation in protocol design and use cases
• Design principles:
  ▪ Protocol instances constructed from building blocks (BB)
  ▪ BBs have common functionality between use cases
  ▪ Try to reuse existing BBs

Note: Some of these ideas are still under discussion among authors. Here we submit them to discussion in NWCRG.
Use Cases

1. NC shim* layer - under TCP, UDP, SSH
2. NC transport, in-net coding
3. NC transport over overlay network
4. NC shim* under tunnel (MPLS, IPsec)
5. Coded TCP (or TCP-like) over disjoint paths
6. NC content dissemination at application layer

(*) Shim: a non-traditional layer, usually between routing and transport

Note: This is not an exhaustive list, but hopefully a large enough set to help identify key building blocks that can be reapplied for different use cases.
Use Case 1: NC Shim Layer – under TCP, UDP, SSH

- Coding: end-end. Passes CC signaling.
- Optional: in-network re-coding.
- Coding nodes determined by: static configuration, routing or control signaling.
- Usage: reliability, similar to source coding.
Use Case 2: NC Transport, In-Network Coding,

- Assisted by multi-path (subgraph) routing
- Usage: reliability, resilience to link and node outage.
- Supports both Unicast and Multicast
Use Case 3: NC Transport over Overlay Network

- Overlay links can be reliable (TCP) or unreliable (UDP).
- Requires both reliability and congestion control functions
- Usage: reliability, resilience to link and node outage, anonymity.
Use Case 4: NC Shim under Tunnel (MPLS, IPsec)

- **Usage**: Provides reliable forwarding under MPLS tunnel
- **Assumes**: configured IP tunnels or routes under NC shim
Use Case 5: Coded TCP (or TCP-like) over Disjoint Paths

- Coding: over all paths
- Congestion control: separate for each path

**Diagram:**
- App
  - MPTCP
    - NC Reliability
      - NC Coding
        - Coded symbols, Rank test
        - Or NC Cong Ctrl
    - TCP
      - NC Rel’
      - Or NC Cong Ctrl
    - IP
  - Tunnel 1 or Netw 1 (eg LTE)
  - IP
- App
  - MPTCP
    - NC Reliability
      - NC Coding
        - Coded symbols, coded pkts
        - Feedback (rank, null space)
    - TCP
      - NC Rel’
      - Or NC Cong Ctrl
    - IP
  - Tunnel 2 or Netw 2 (eg, WiFi)
  - IP
Some terminology

- **input flow terminology**
  - an input flow enters a NC protocol instance
  - an input flow contains **input packets**
    - a packet may be a UDP datagram, an IP datagram, a frame, an application data unit, a file slice, etc.
  - an input packet contains **input symbol(s)**
    - plus protocol headers, control information, etc.
    - packet/symbol mapping can be $1 \leftrightarrow 1$ or $1 \leftrightarrow \text{multiple}$ (not assumed to be frequent) or multiple $\leftrightarrow 1$ (if fragmented, when needed by the use-case)
  - an input symbol can be a **source** symbol or a **repair** symbol (encoded one or more times)
Some terminology… (cont’)

- **output flow terminology**
  - output flow/packet are similar
  - an output symbol is an *encoding symbol*, i.e., either a *source* symbol (at a decoder or in case of a systematic FEC) or a *repair* symbol

- **NB: “encoding symbol” definitions in current RFC**
  - [RFC 6363]
    - Encoding Symbol: Unit of data generated by the encoding process. With systematic codes, source symbols are part of the encoding symbols.
  - [RFC 5052]
    - Encoding symbol: A source symbol or a repair symbol.
Close-up on the FEC building block

- let’s see the FEC BB terminology

**NC Protocol Instantiation**

- **mapping** to BB’s input symbols (can be 1⇒1 or more complex)
- Input flow(s) contain input packets
- Output flow(s) contain output packets

- Input symbols
- Output symbols
An example

Let’s consider an RLC FEC BB

It’s just an example, other FEC BB will be considered in future

Example: encoding side

Example: basic RLC BB

current encoding window

Element i is

\[ c_0 \cdot \text{symb}_0[i] + \ldots + c_3 \cdot \text{symb}_3[i] \]

Element in \( \text{FF}(2^m) \) (e.g. byte if \( \text{FF}(2^8) \))
More about FEC terminology

- FEC scheme (fully specified, see RFC 5052)

FEC Scheme

\[ \text{FEC Scheme} = \{ \text{identifier} + \text{code specifications} + \text{signaling} \} \]

- each scheme is uniquely identified (IANA registry)
  - FEC Encoding ID ex. 5 for Reed-Sol. over FF(2^8) in the context of RMT

- all the code details are specified non ambiguously
  - interoperability is a MUST

- signaling enables encoder/decoder synchronization, for a given object transfer
More about FEC terminology… (cont’)

- yes, we need a FEC Encoding ID
  - for instance:
    - FEC Encoding ID 100  binary RLC
    - FEC Encoding ID 101  RLC over GF(2^4)
    - FEC Encoding ID 102  RLC over GF(2^8)
    - FEC Encoding ID 103  Structured RLC
    - FEC Encoding ID 104  another FEC solution…

  - NB: ID 100 can also refer to RLC over GF(2^m), where m is carried in the signaling part… It works too!

  - this FEC Encoding ID points to a specific FEC BB and a specific way of doing signaling
    - so that a NC instance knows exactly how to process it
Network Coding System Decomposition
Network Coding Functional Areas & Building Blocks 1/3

• NC Coding – all coding operations
  ▪ E.g., encoding, decoding, test for “innovative”, rank, null space
  ▪ Using operations such as finite field and linear transformations

• NC Reliability – data and control to support reliable transfer
  ▪ Includes reliability logic (end-to-end and/or hop-by-hop), coding vectors, feedback.
  ▪ May be subdivided in FEC BB + coefficient BB + header BB
NC Congestion Control – controls transmission rates
- Flavors: unicast CC, multicast CC, subgraph CC
- Should try to use algorithms developed in other WGs when possible, such as TCP-Friendly based on the PFTK formula [1], as in NORM [2]
- Subgraph CC - most likely new


Network Coding Functional Areas & Building Blocks 3/3

• Multi-path routing, multi-path forwarding
  ▪ Related to NC reliability BB through: splitting ratios (fwd factors), up/down neighbors, link quality
  ▪ Most likely: augment existing routing and fwd protocols

• Security – First option: rely on existing solutions
  ▪ Unless creating a new security protocol with NC as essential part
  ▪ Can do: pollution detection at the packet level, without decoding, or detection and correction at a layer that decodes
Conclusions, Issues and Open Questions

- Modular reuse approach based on BB seems to work: all use cases presented can be built using a very small number of BBs.
- Architecture of use cases needs (a lot of) work to be mapped/integrated into the IETF layers/areas.
- NC Coding and NC Reliability BBs – are core elements.
- Congestion Control – can use existing algorithms
  - CC for general subgraphs does not exist.
  - NC under TCP can raise questions about fairness – some answers exist. Need to clarify.