About Network Coding terminology and concepts

Quick & dirty draft for discussion purposes!

Vincent Roca (Inria, France)

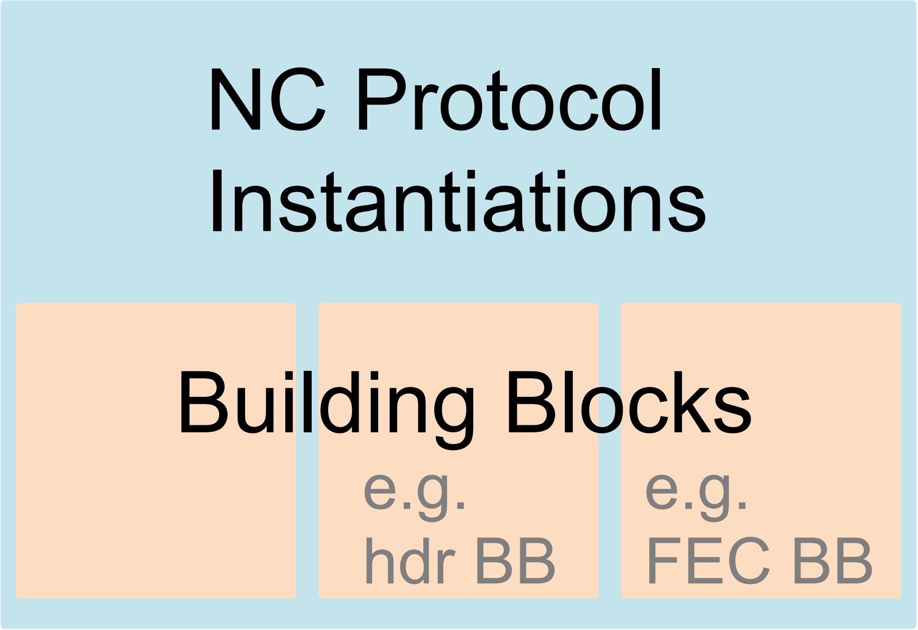
IETF89, NWCRG meeting

March 6th, 2014, London

NC terminology and architecture

Basic idea (as in RMT)

- assemble BBs and create a NC protocol instantiation
  - ❍protocol = { building blocks (specialized if needed) + some glue }
  - ❍working solution
- "building block" (BB) approach
  - ❍focused, reusable components

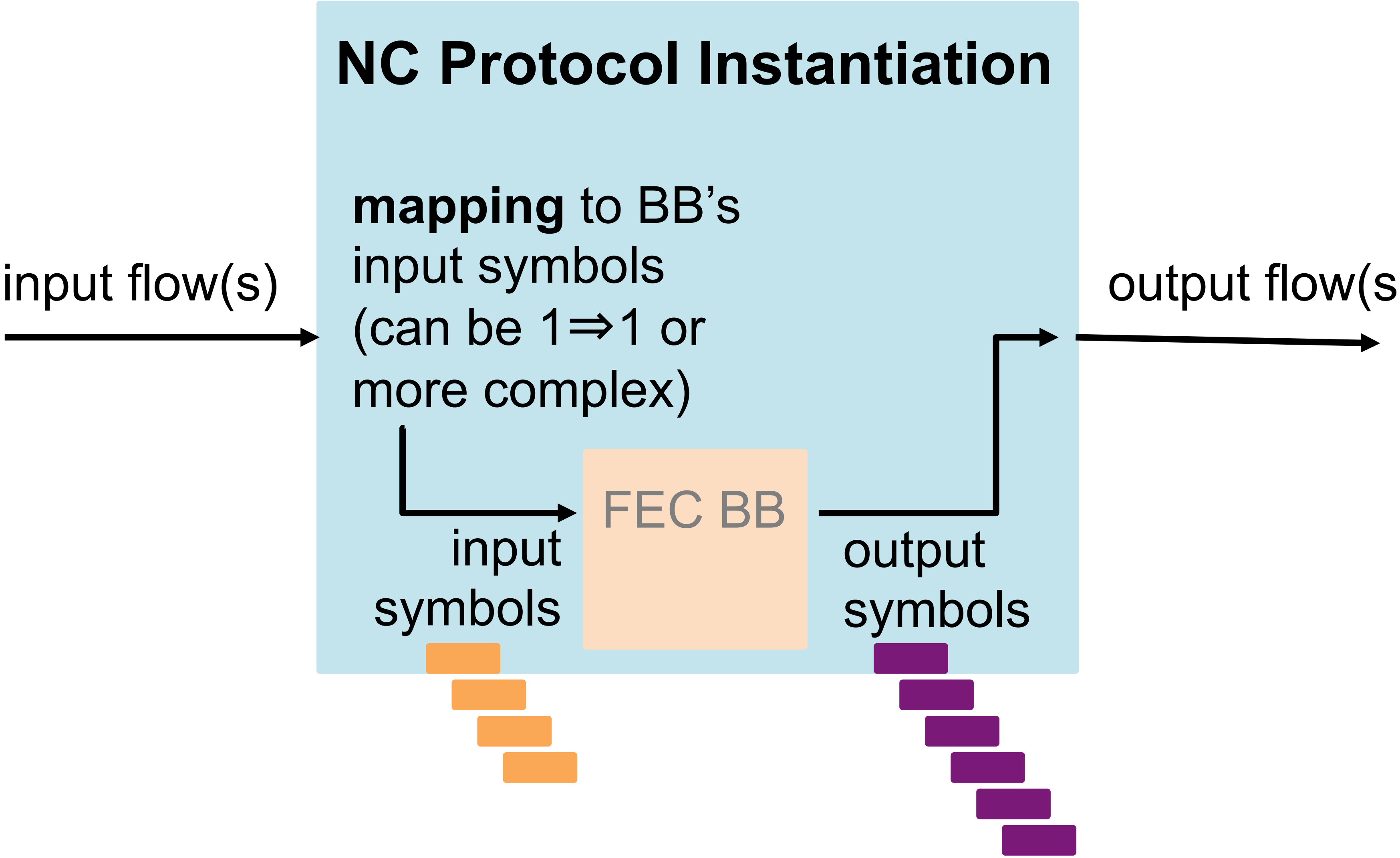


NB: what I'm calling **FEC BB** might be called **Coding BB**

NB: at RMT, there is an additional level, on top of PI, namely "application" like FLUTE or FCAST that further instantiate a PI to turn it into a practical solution

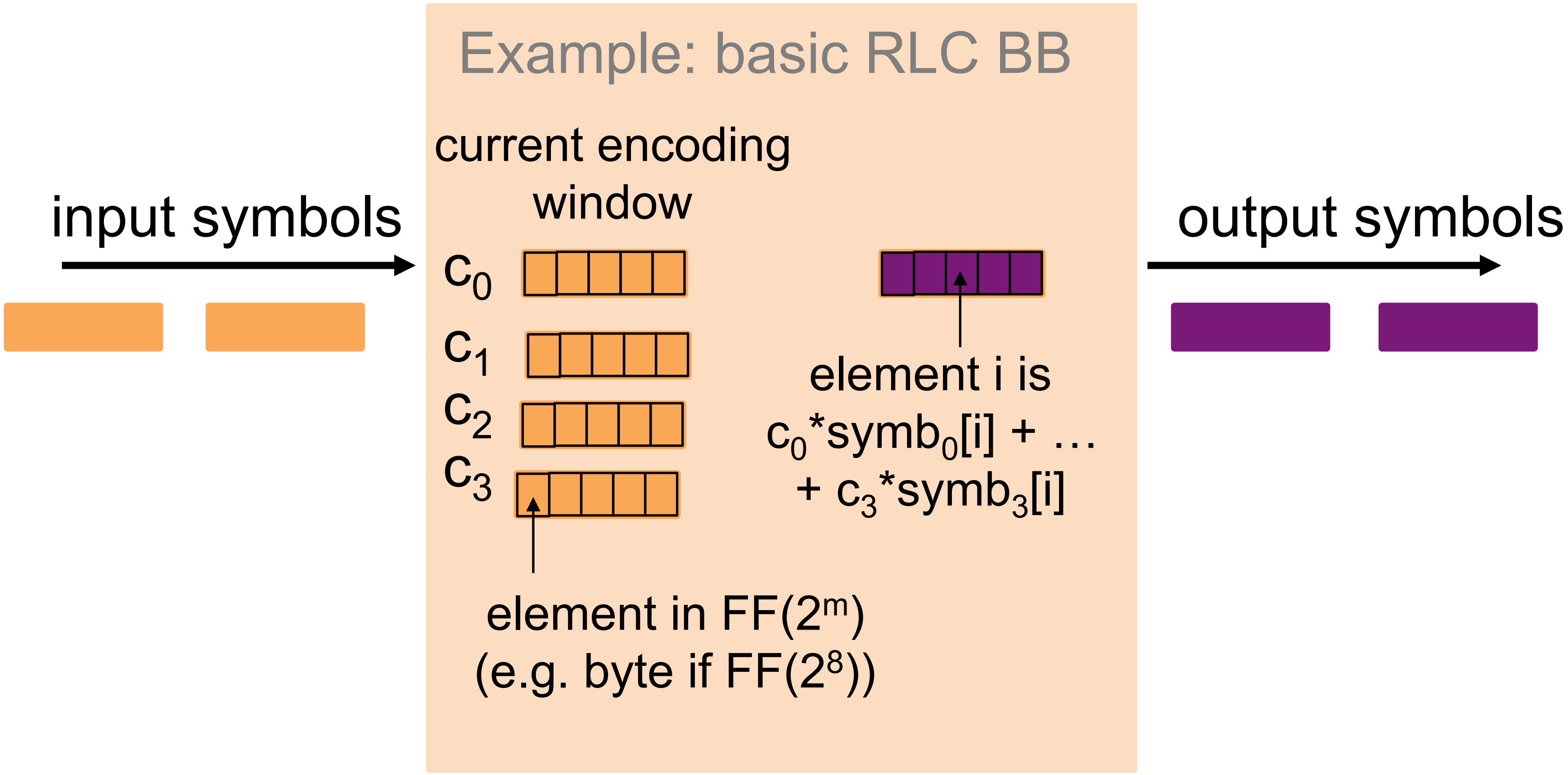
FEC BB and terminology

- let's see the FEC (e.g. RLC) BB terminology


NC Protocol Instantiation
input flow(s)
output flow(s)
mapping to BB's input symbols (can be 1⇒1 or more complex)
FEC BB
input symbols
output symbols

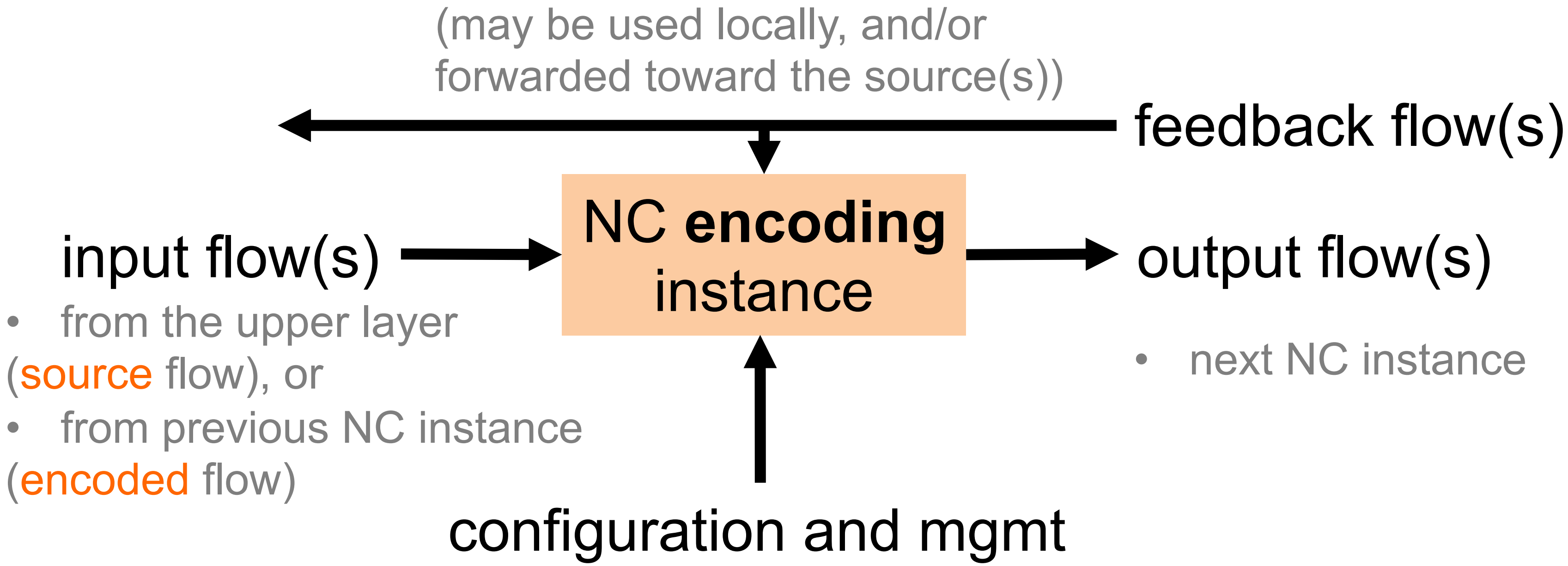
FEC BB and terminology… (cont')

- let's look further at the FEC (e.g. RLC) BB
  - ❍example: **encoding side**
  - ❍**elements in the FF can be considered if need be**


Example: basic RLC BB
current encoding window
input symbols
output symbols
c_0
c_1
c_2
c_3
element i is c_0*symb$_0$[i] + … + c_3*symb$_3$[i]
element in FF(2^m) (e.g. byte if FF(2^8))

Architecture (high level view)

- encoding side (independently of the layer where it's applied)

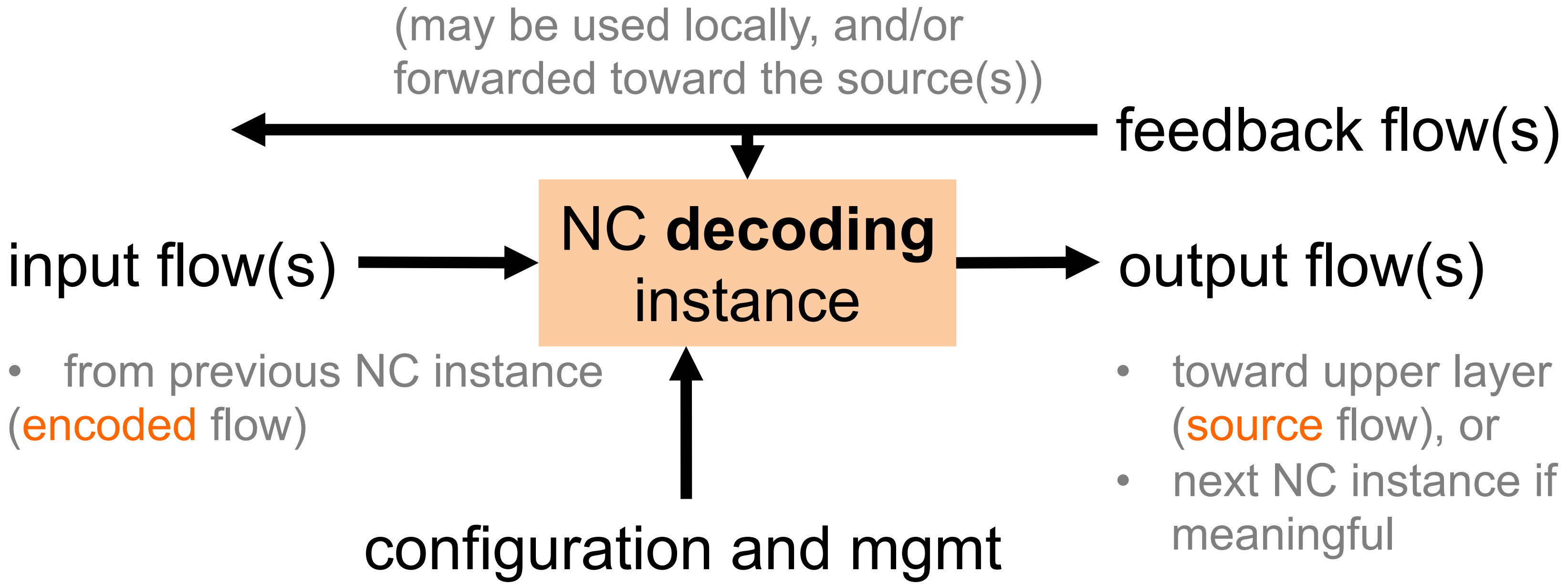


- from the upper layer (source flow), or
- from previous NC instance (encoded flow)

- next NC instance

Architecture (high level view)

- decoding side (independently of the layer where it's applied)



(may be used locally, and/or forwarded toward the source(s))

feedback flow(s)

input flow(s)

NC **decoding** instance

output flow(s)

- from previous NC instance (encoded flow)

configuration and mgmt

- toward upper layer (source flow), or
- next NC instance if meaningful

Additional missing important terminology

- FEC scheme (fully specified, see RFC 5052)

FEC Scheme

=

{identifier + code specifications + 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

Additional missing imp. terminology (cont')

- yes, we need it!
  - for instance
    - **FEC Encoding ID 100 refers to binary RLC**
    - **FEC Encoding ID 101 refers to RLC over GF(2^4)**
    - **FEC Encoding ID 102 refers to RLC over GF(2^8)**
    - **FEC Encoding ID 103 refers to our proposed SRLC**
    - **FEC Encoding ID 104 refers to another Structured RLC**
    - **…**
    - **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
    - **all NC instances know exactly what to do**

Examples of NC Building Blocks

Non exhaustive BB list

name	description
Finite Field BB	specify how FF computation is performed, how elements are managed in symbols, etc.
coefficient list encoding BB	• can be explicit: full vector, or compressed list (e.g. Run Length Encoding) • can be "implicit": as a tuple {function; value} that gene-rates the coefficients (e.g. PRNG + seed + algorithm)
FEC BB	actual FEC solution (may reuse the FF and coefficient list encoding BBs)
header BB	*Q: is a generic header feasible?* In any case, it MUST include a generic header extension mechanism (e.g. ALC EXT_...)
congestion control BB	especially with NC at transport level, as the main protocol
security BB(s)	If done within the NC instance. *Q: is it the right approach? Perhaps for some NC-specific security services… TBD* Otherwise we re-use existing security solutions… Involves all the basic services (authentication, integrity, non repudiation, anti-replay, confidentiality, etc.)
…	…