Consiglio Nazionale delle Ricerche
Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni

2nd Italian Workshop on Embedded Systems
IWES 2017
September 7-8, 2017 — Rome, Italy

CAN with eXtensible in-frame Reply: a Survey

G. Cena, I. Cibrario Bertolotti, T. Hu and A. Valenzano
CNR-IEIIT (Torino)

CAN Timeline

- 1986: CAN (Kiencke U. et Al., "*Automotive Serial Controller Area Network*," SAE Technical Paper 860391)
  - CAN was *first* presented
- 1991: CAN 2.0B (BOSCH, CAN Specification 2.0)
  - Identifier field enlarged from 11b to 28b (*extended* IDs): the number of messages grows from 2048 to more than *half a billion*
- 2001: TTCAN (Fuehrer T. et Al., "*Time Triggered CAN*," SAE Technical Paper 2001-01-0073)
  - *Time-Triggered* communication paradigm on CAN
- 2011: CAN FD (BOSCH, CAN with Flexible DataRate 1.1)
  - Maximum payload size enlarged from 8B to 64B (*oversizing*)
  - Bit rate can be increased in the data phase (*overclocking*)

CAN with eXtensible in-frame Reply

- Every new version of CAN is *unable to coexist* with controllers complying with *previous protocol* generations
  - Unless *new features* are *not exploited* (quite a limitation!)
- **Is it possible to enhance CAN further?**
  - Basic requirement: *full coexistence* with legacy CAN controllers and devices must be preserved
- **Yes! The solution is CAN XR**
- This can be done by exploiting:
  - *In-bit-time detection:* at any time, in CAN, every node in the network virtually sees the same bus level (either *dominant* or *recessive*)
  - *In-frame reply*: unlike remote frames, a reply is *immediately* sent on the bus when specific conditions are met (from VAN)

CAN XR transactions

- Data exchange in CAN XR takes place in *transactions*
- With respect to *any given* transaction, two kind of nodes are defined with different roles:
  - *Initiators* (one or more): take care of *starting* transactions
  - *Followers* (any number, including none): deal with *data exchanges*
- Initiator:
  - Carries out arbitration and sends the control field (*header*)
  - Supervises the transaction and concludes the related frame (*trailer*)
- Followers:
  - *Responders*: reply to the transaction's header by filling the data field
  - *Consumers*: nodes interested in data included in a transaction

Initiating transactions

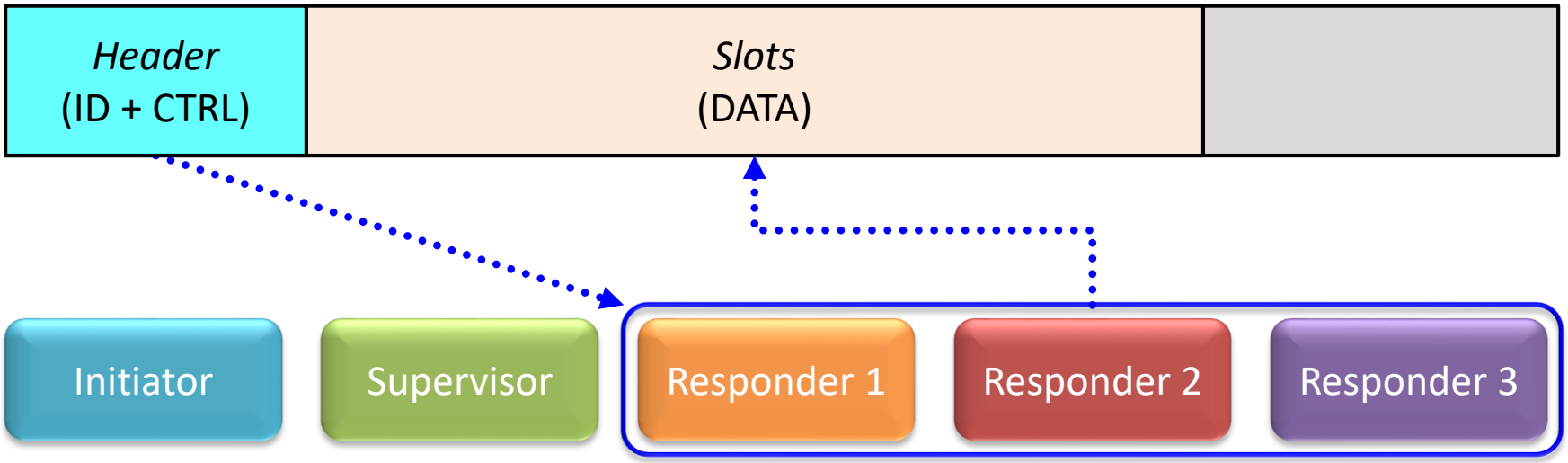
- Each transaction is *initiated* by the relevant initiator
  - A *new service* is defined which only sends the *header* on the bus
  - Transactions are *distinguished* using the CAN identifier field (ID)
  - The ID field is also used to discriminate between conventional CAN frames and those bearing transactions (*CAN XR frames*)

Initiating transactions (II)

- Variations to the *basic* approach are possible
- *Multiple initiators*:
  - More than one node is allowed to initiate a specific transaction
  - A *group* of CAN IDs with a common *prefix* is exploited
  - A suitable *reception mask* is defined on the related followers
  - Resemble *backup time masters* in TTCAN
  - Increase flexibility and reliability
- *Implicit initiators*:
  - Multiple initiators chosen as a *subset* of the followers
  - There is no node acting as a *pure* initiator
  - As soon as one responder starts transmitting all the others *follow*
  - Decrease costs and achieve spatial data coherence

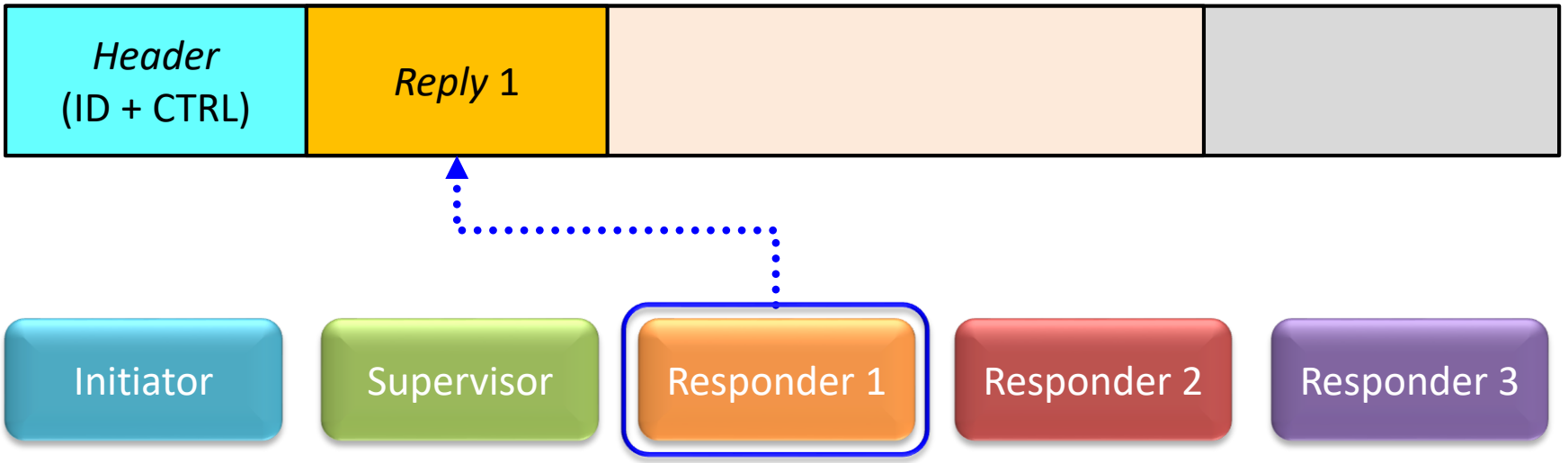
Taking part to transactions

- Followers *take part* to transactions they are interested in
  - They *sense* the bus looking for transactions (headers are sought)
  - H/W message filtering on ID is *mandatorily* required for *responders*
  - This is because *insertion* of replies has to be done *on-the-fly* without disrupting the bit sequence on the bus



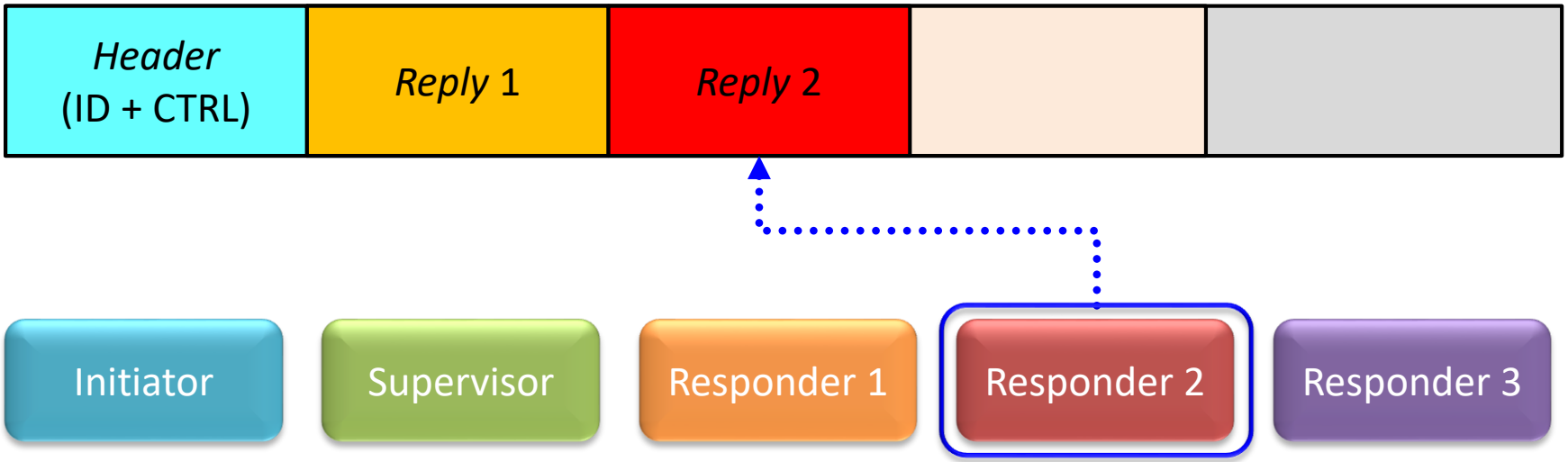
Taking part to transactions (II)

- When a *relevant* header is found
  - Specific *portions* of the CAN frame's data field (*slots*) are separately filled/acquired
  - Each responder *replies* by sending its stored data in the relevant slot
  - Each consumer *reads in* data from relevant slots



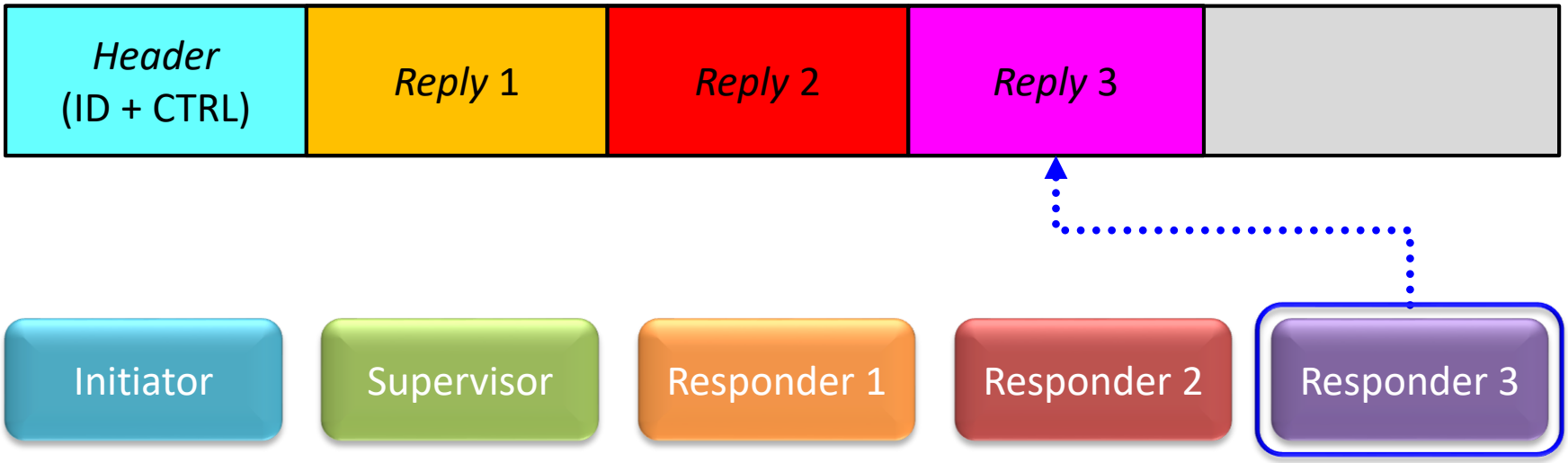
Taking part to transactions (III)

- Different kinds of slots may be defined
  - E.g., *static* vs. *dynamic*
  - To enable particular behavior
  - To implement specific network services
  - To offer increased flexibility (protocol *extensibility*)



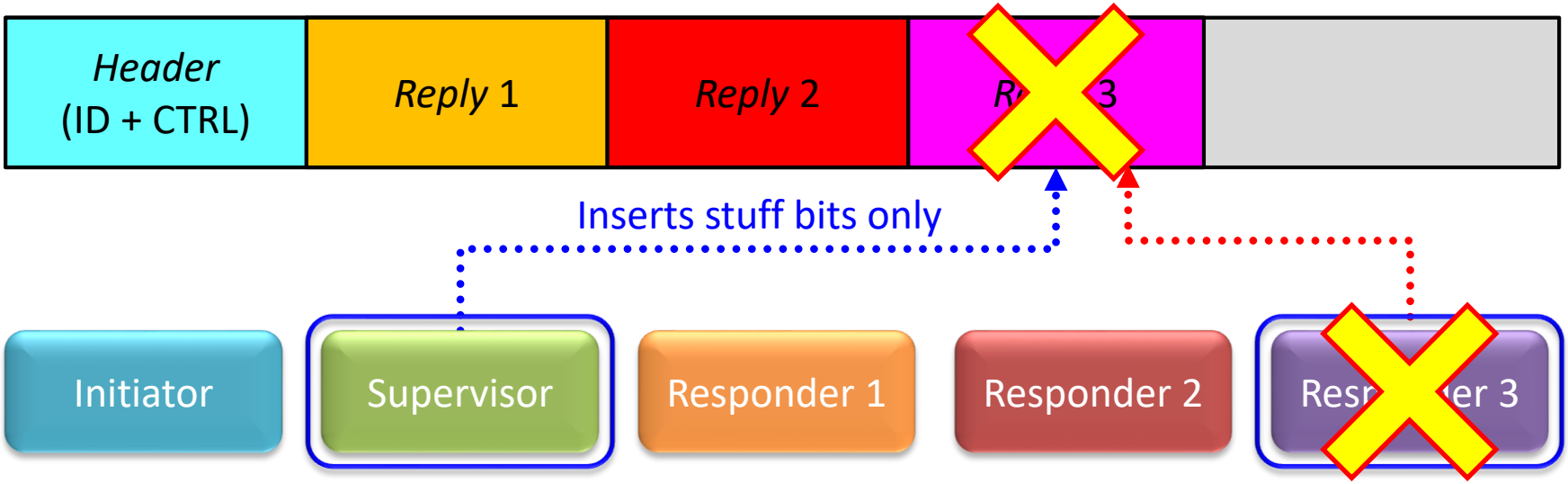
Taking part to transactions (IV)

- Slots are *configured* in advance in relevant followers
  - *Static* slots are defined in terms of *initial position* and *size* (in bits) in the data field
  - *Dynamic* slots are defined in term of their *relative order* in a statically configured part of the data field



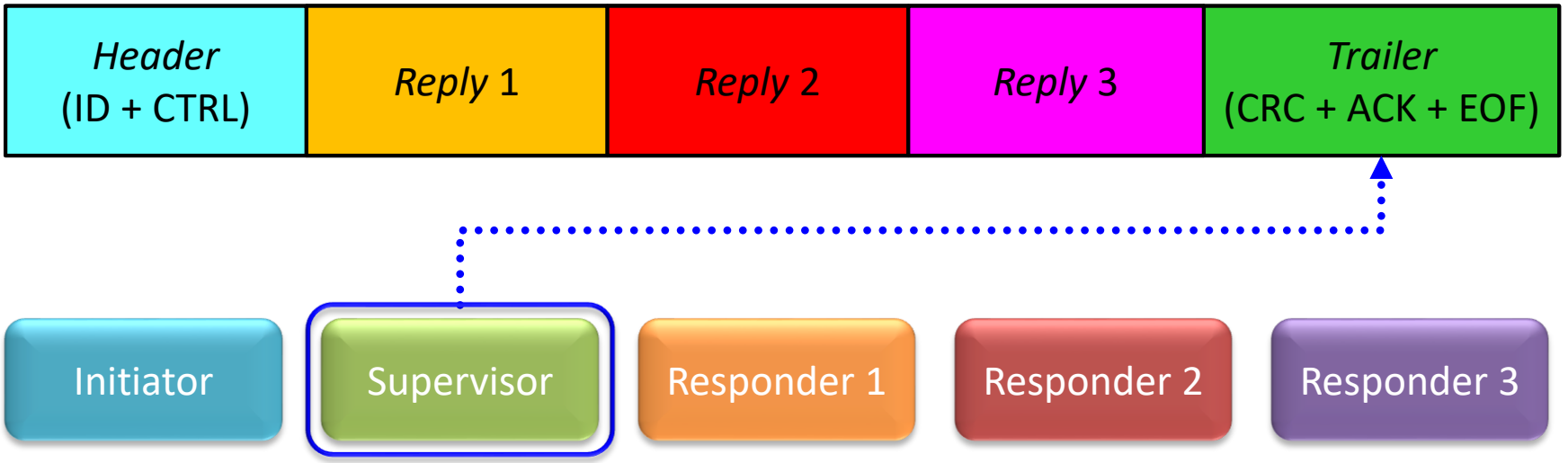
Supervising transactions

- Each transaction must be *supervised*
  - If a responder is *not active* its slot remains at recessive level
  - This event must be dealt with to prevent *bit stuffing* errors
  - Part of the *supervisor* role is to *insert* stuff bits when needed to preserve frame correctness in the case of *missing* replies



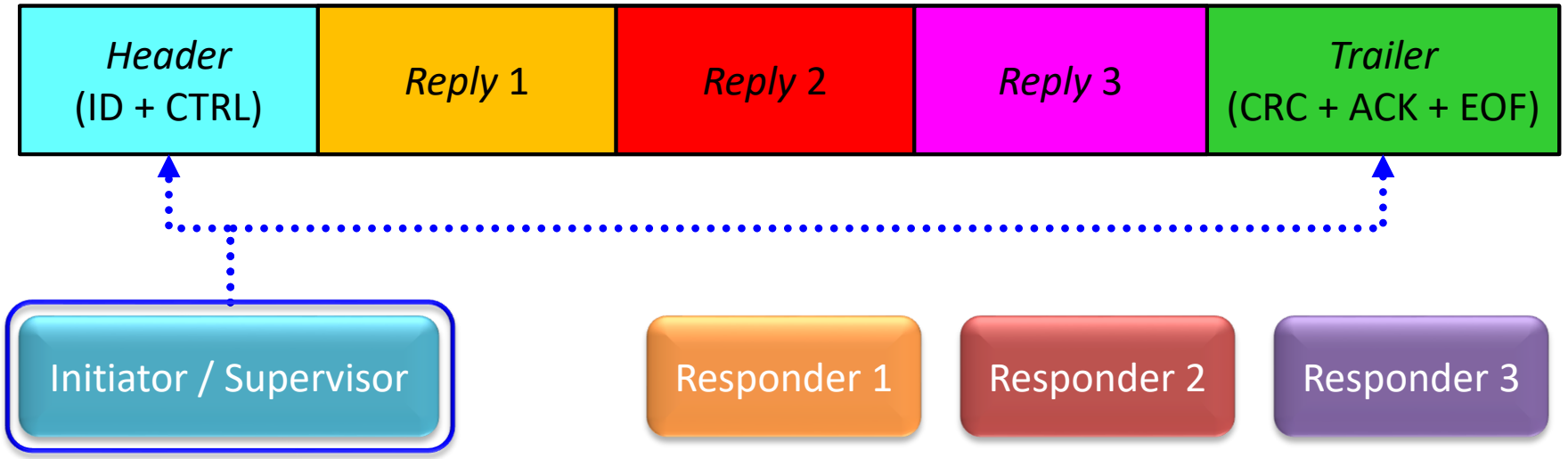
Supervising transactions (II)

- Each transaction must be *completed*
  - Another duty of the *supervisor*
  - The supervisor deals with CRC, ACK, and EOF fields (*trailer*)
  - Ensures that *error-free* transactions *resemble* well-formed CAN frames



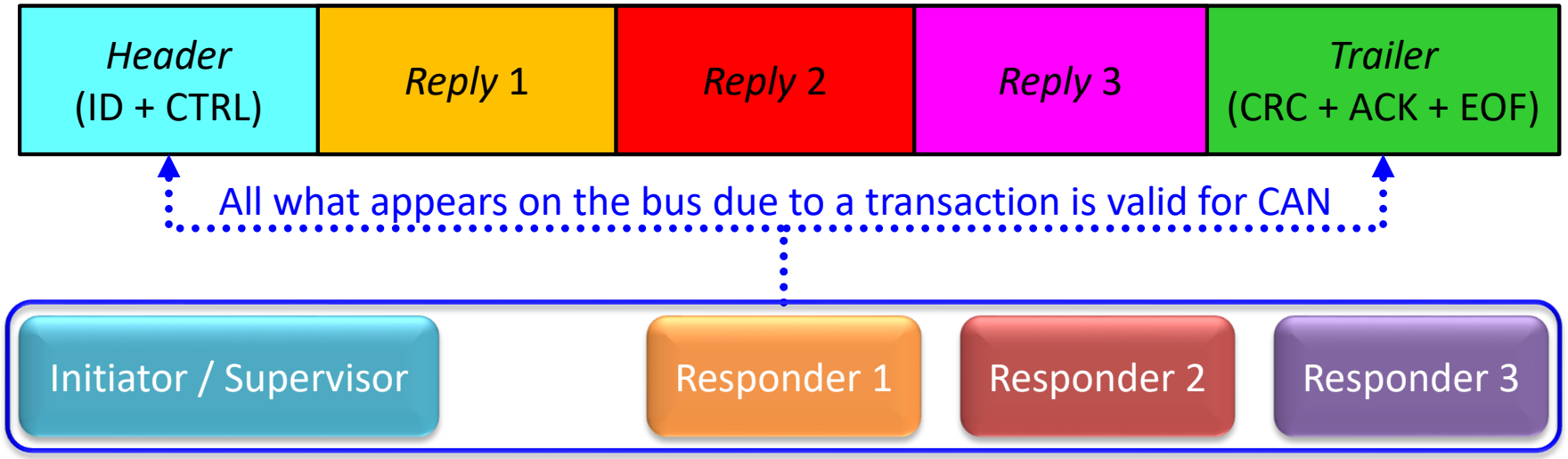
Supervising transactions (III)

- The best option is that *initiator* and *supervisor* for any given transaction *coincide*
  - Most *straightforward* (and simple) choice
  - All initiated transactions are supervised (almost) *for sure*
  - Highest *reliability* (single point of failure)



Supervising transactions (IV)

- Overall, CAN XR is conceived so that
  - The *bit sequence* produced on the bus by the *initiator/supervisor* and *responders* is *indistinguishable* from a conventional CAN frame (either Classical or FD, Base or Extended)
  - *Coexistence* with non-XR legacy CAN controllers is preserved

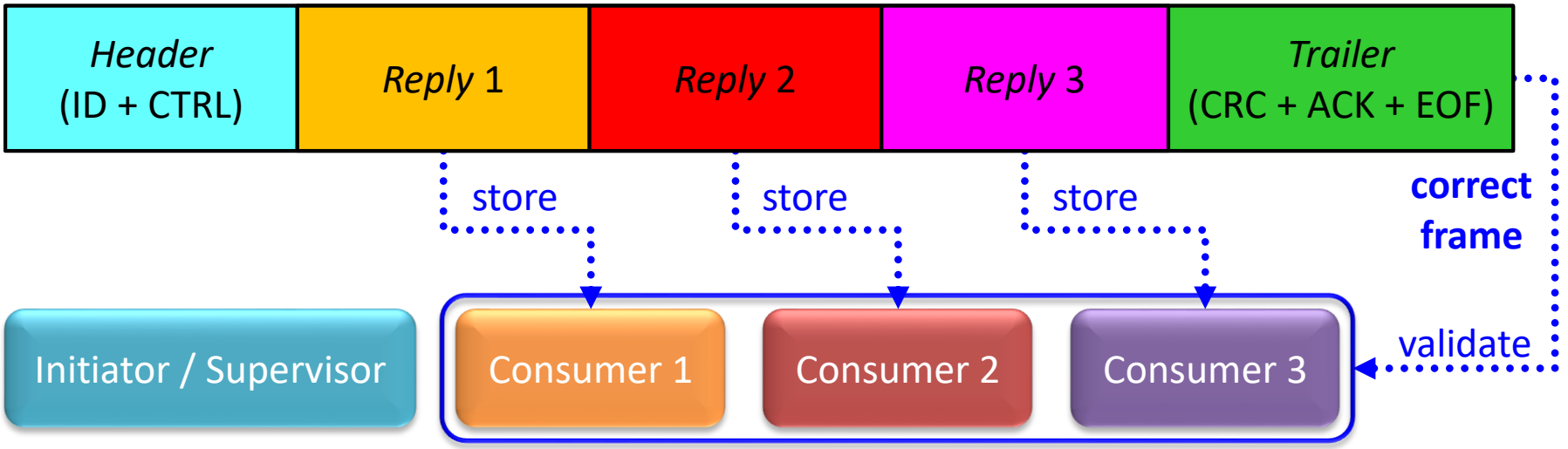


Summing up: supervisor duties

1. *Decoding* the bit stream on the bus during the whole data field and *inserting* stuff bits when needed
   - Position and value of stuff bits inserted by *active responders* and the *supervisor* coincide (they *overlap* seamlessly)
   - Should some responder not reply (due to temporary *unavailability*) the transaction does not get corrupted
2. Dealing with the frame *trailer* (after the data field)
   - *Generating* the CRC from what is read on the bus and *appending* it to the data field
   - *Managing* the ACK slot and dealing with the related ACK *errors*
   - Dealing with the EOF field

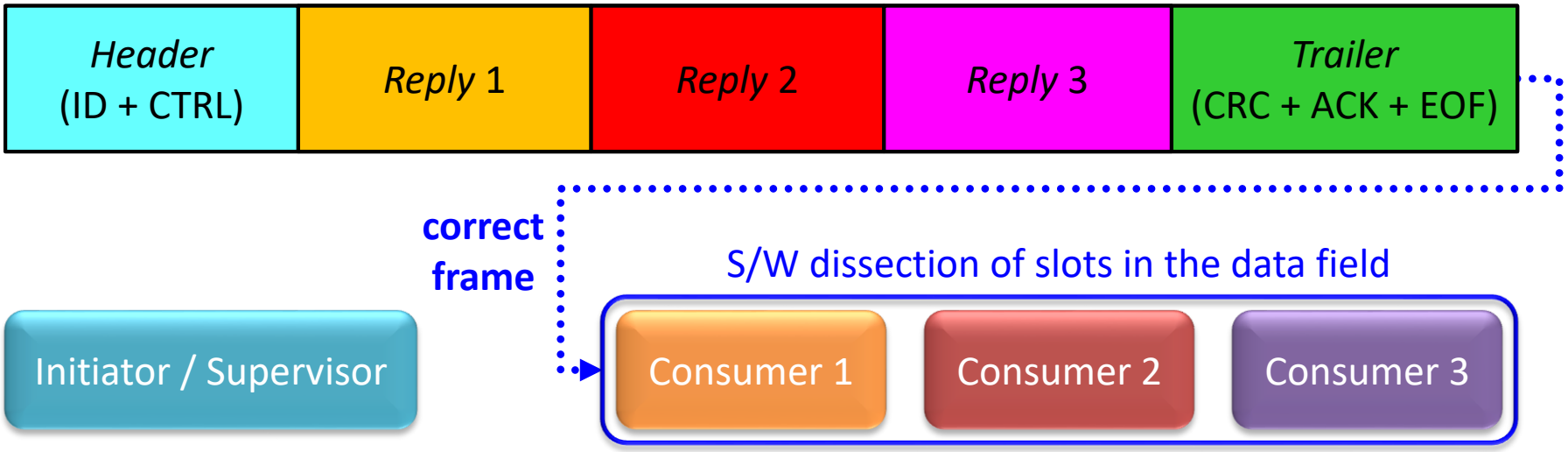
Consuming data in transactions

- Consumers interested in *specific* pieces of data *read* them in
  - Message filtering on ID is used to *single out* relevant transactions
  - Every node in the network carries out *error detection* as per CAN rules
  - If no errors occurred the values of the relevant slots are *stored* locally



Consuming data in transactions (II)

- Consumers are *not required* to rely on XR controllers
  - Conventional (*non-XR*) controllers can be adopted as well
  - The data field is first read in *as a whole* and then *dissected* in S/W
  - Performance of S/W solutions is *lower* than H/W solutions
  - They also have larger *local storage* requirements



Static Slots

- *Exclusive* slots:
  - *Exactly one* responder is allowed to reply in the slot
  - Mostly resemble data transmission in CAN but support *data gathering*
- *Shared* slots:
  - *Any number* of responders is allowed to reply in the slot
  - Bit monitoring must be *disabled* for recessive bits
  - A network-wide *bit-wise wired-AND* is carried out among responders
- *Arbitrating* slots:
  - *Any number* of responders is allowed to reply in the slot
  - Resemble shared slots but a responder *stops* transmitting as soon as it loses arbitration (dominant level sensed while writing a recessive bit)
  - The network-wide *minimum* among values of replies is obtained

Dynamic Slots

- A *variable* number of them may fit in a *dynamic segment*
  - Each *reply* is prepended with its *size*
  - *Slot Length Code* (SLC): same encoding as DLC (on 4b)
  - Permits *on-the-fly dissection* of the dynamic segment
- Followers obey a *linear arbitration* access procedure
  - A *slot counter* (SC) is set to 0 at the beginning of the dynamic segment and increased by one on every new slot
  - Each piece of data is assigned its unique *slot index* (SI)
  - When SC=SI the reply can be written/read
  - Missing replies generate a *minislot* (SLC=1111)
  - If the remaining room in the dynamic segment is not enough for the reply a *deferral notice* (SLC=1110) is sent in its place

Examples of CAN XR Applications

- *Combined message*
  - Communication efficiency higher than CAN FD for small data packages
- *Distributed key generation*
  - A. Mueller and T. Lothspeich, "*Plug-and-secure communication for CAN*," Proc. Intl. CAN Conference (iCC), Oct. 2015, pp. 06-6–06-14
- *Min-Max discovery*
  - Minimum can be discovered with a single CAN exchange
- *Event notification*
  - May possibly rely on shared slots for multi-source events
- *Distributed consensus*
  - Exploits transaction atomicity and robust CAN error detection

Conclusions

- **CAN with eXtensible in-frame Reply (CAN XR)**
  - Increases communication performance thanks to *data gathering*
  - Achieves a number of *distributed atomic* network functions
  - Yet retaining complete backward *compatibility* and *interoperability* with existing CAN/CAN FD devices
- **We did it!**
  - Using a purposely developed *software-defined CAN controller* (SDCC)
  - Protocol *correctness* has been *verified* as well as *interoperability* with real CAN devices
  - G. Cena, I. Cibrario Bertolotti, T. Hu, A Valenzano, "*CAN with eXtensible in-frame Reply: Protocol Definition and Prototype Implementation*", IEEE Transactions on Industrial Informatics, 2017, Early Access

Thanks for your attention

Any question?