Token Cell Routing Data Plane Concepts

draft-bcx-rtgwg-tcr-00

Stewart Bryant (Institute for Communication Systems, University of Surrey)

Alexander Clemm (Futurewei US)

draft-bcx-rtgwg-tcr-00

Motivation and requirements to address

- New network demands stress existing dataplane protocols
  - Collection of telemetry
  - Path guidance (including reroute protection)
  - Guaranteed SLOs, e.g. maximum (and minimum!) latency
  - Programmability (including parametrizable function)
  - Security, proof-of-transit, extensibility etc
- Requirements (excerpt)
  - Allow support for multiple features concurrently
    e.g. protected segment path with distinct telemetry collection per segment and latency SLO
  - Security
    e.g. authenticate metadata added by intermediate nodes
  - Extensibility & permissionless innovation
    e.g. add new features without need for protocol extensions

Basic TCR Idea

- Functionality encoded as lightly structured “tokens” (referred to as token cells) with multiple tokens in a packet.
  - Apply longest prefix match engines to invoke code points
  - Tokens can be combined for more elaborate functionality (and integration of multiple functions / use cases)
  - Stacking available for per-segment, per-node behavior where needed
- Inherent extensibility and customizabilty
  - Combination of token cells (think Lego)
  - Token cell parametrization
  - Extensible Token Cell Engine (for new token cell types)
- Complemented with
  - Metadata support (readable **and** writeable)
  - Advanced security support (differentiated signing)
  - Processing workflow framework (parallelization, multi-token pipelining)

Key differentiators

- Specify multiple actions at a hop in a compact way
- Allow for non-linear parsing of packets
  - Combine multiple actions without need to rewrite packets
  - Clearly defined interdependencies facilitate performance optimization
  - New features require new procedures to be installed along the path with pointers to the procedures in the packet – like MPLS with arbitrary sized LSEs
- Programming packet behavior supported at multiple levels:
  - Construct a packet from multiple tokens to combine actions to define behavior
  - Parametrization of tokens, as applicable and where needed
  - Extension with new token types is possible –
    operators can install new code points to be invoked
- Special token types to enhance security (e.g. authentication of metadata), for scratchpads (writeable metadata by intermediate nodes), conditional behavior, etc
- TCR may be applied as a packet design in its own right, or as a method of describing advanced functions to extend the capability of an existing protocol.

TCR packet structure: Just a series of token cells

Identify as TCR Packet
(could possibly be a token cell itself)

Token cells are “stem cells”
Represent a “unit of processing”…
Semantics and how to process will be superimposed by token cell type

What, no separate payload?

- An optional approach - can use classical design but:
- If carried as part of separate payload token allows for different payload delivery semantics (e.g. application of network codings, …)
- Payload-less packets can be used for signaling and control purposes

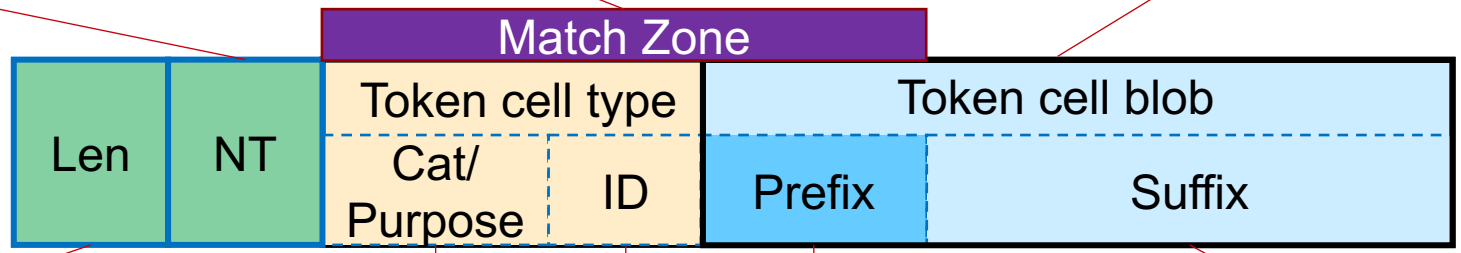
Token cell structure

Notes:
- *Token cell disposition can be handled in different ways, for example:*
  - *Specify as preamble (e.g. "Disposition Action" next to "NT")*
  - *And/or in a complete separate disposition token cell*
- *Can use different encodings for token cell field, e.g. length:*
  e.g. linear fixed len vs var len

Match zone is of fixed length
Can be matched/processed in one step

- Any structure within the token cell blob
  (such as subfields, parameters) is up for token cell type to define
- Blob contents within the prefix zone can be part of
  token cell engine lookup

Next token cell to serially process, if any



Token cells may vary in length
May (or may not) be fixed per token type

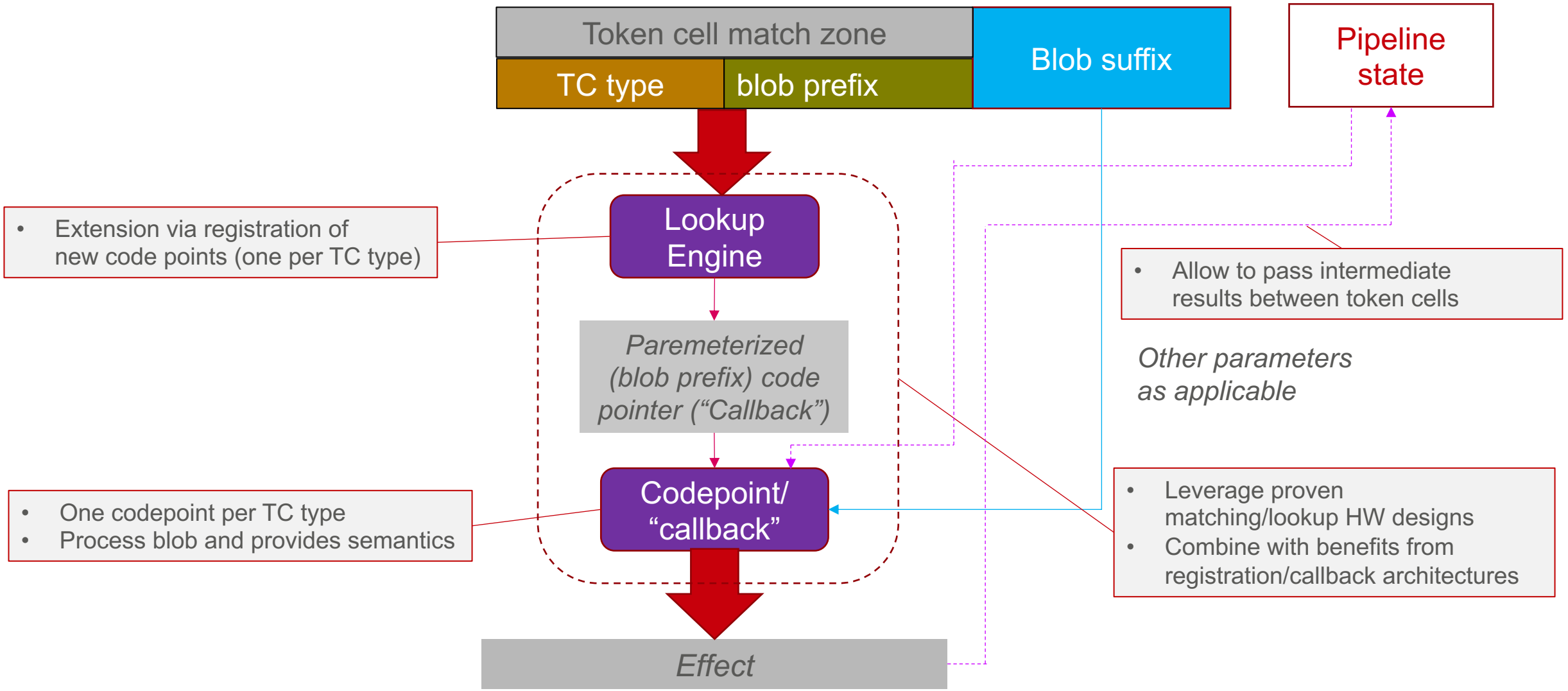
- Portion of the blob that exceeds "match zone"
- Can be processed by matched code point
  (but is not part of token cell engine lookup)

- ID prefix (not a fixed or even fixed length field)
- Allows for token cell type categorization

- Built in set of IDs
- Can be extended by users
- Registered
- Not a fixed length field but a convention
  (token type as a prefix for the match zone)

- Portion of the token cell blob that is
  subject to lookup
- Lookup includes token cell type and
  parameters/data in blob prefix

Token Cell Processing: Individual Token Cells



Token Cell Type Categories

- Forwarding
  - Token type per addressing type
- Metadata
- Scratchpad (writeable metadata, separately secured)
- Security (incl. signed metadata from multiple hops)
- Manifest & rendezvous (concurrent processing)
- Disposition
- Directives
  - SLOs: e.g. latency objectives
  - Special purpose directives: LBF and special QoS actions; Telemetry & IOAM Collection, Flow Profiling, …
  - General purpose directives: templates for condition – action – parameters
- Conditionals
  - Assess dynamic condition prior to continued processing
- Other: anything else

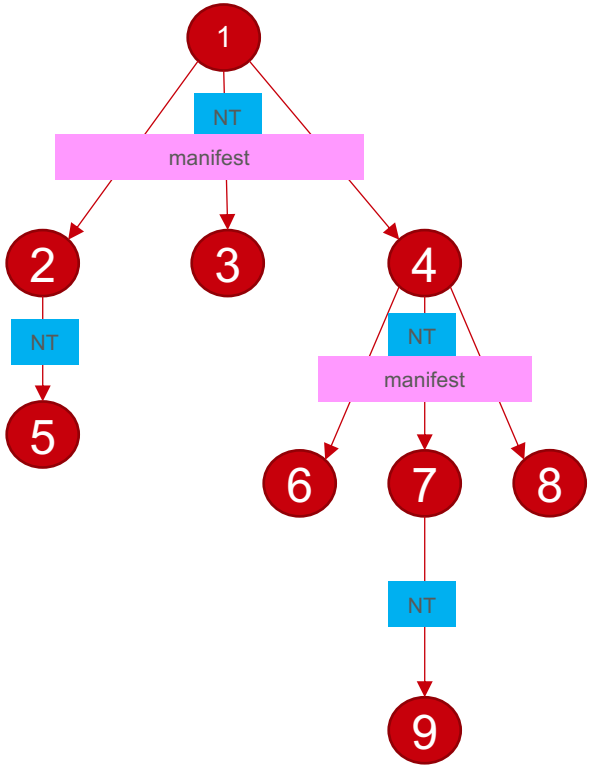
- Token cell types fall into categories, “subtype” the category
- New types can be registered; category prefix of TC type prefix of match zone
- Each TC type has its own semantics, implemented via respective code points

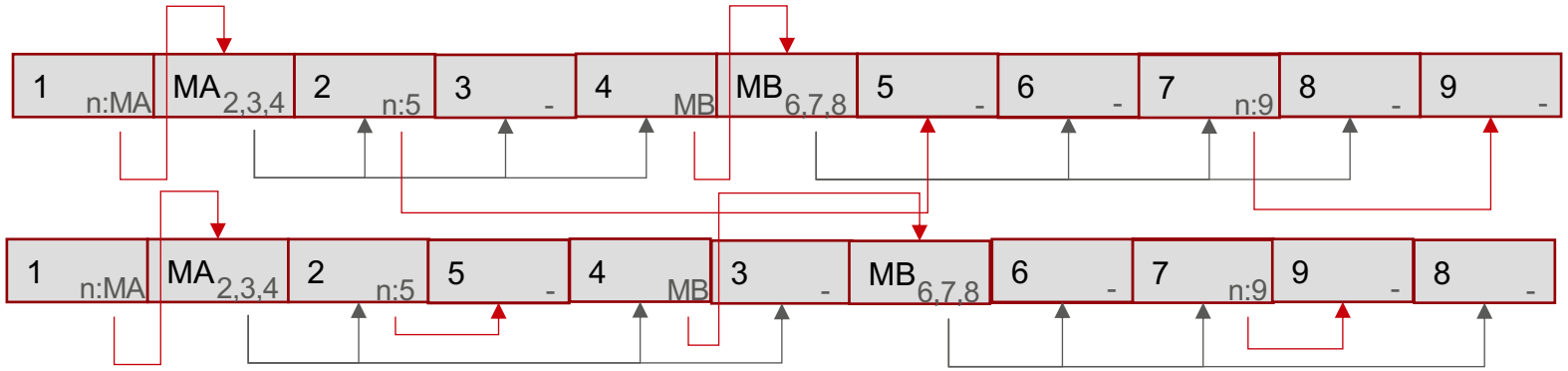
Serialization and Parallelization Framework

- First token cell is always processed
- Need to distinguish which token cells to process and whether to serialize
  - Some TCs are just “carried” – scratchpads and metadata accessed by other token cells
  - Some TCs can be processed in parallel – optimize performance
  - Some TCs need to be serialized – interdependencies
- Serialization:
  - Token cell include a “Text TC” field: Null – stop, next – “goto”
  - Only forward references (avoid loops)
- Parallelization:
  - Addition of manifest: indicate parallel “start points”
  - Manifest is just another token – introduce wherever parallelization is needed

Processing is done when all “reachable” TCs have been processed

Packet permutations can be accommodated; TCs do not need to be arranged in a stack

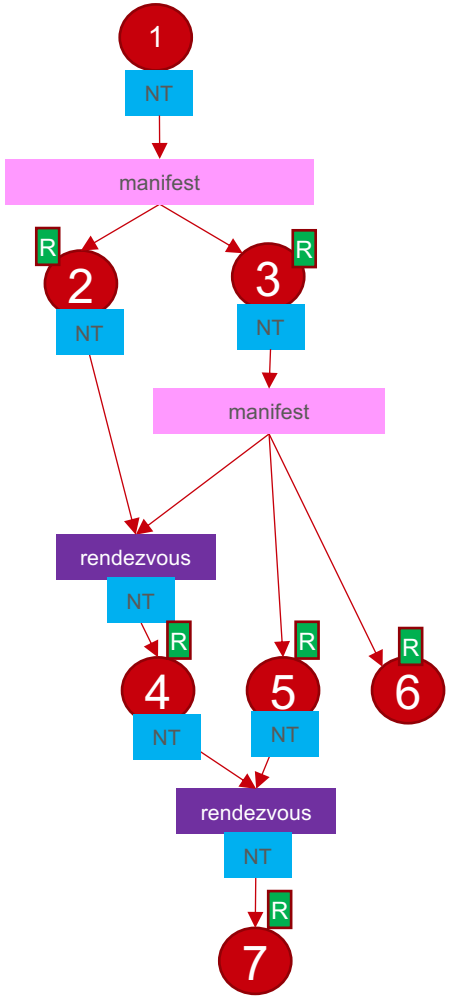


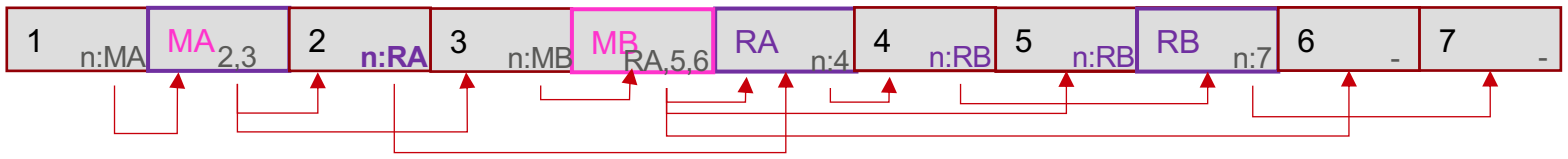


Serialization and Parallelization Framework (contd.)

Note: we do not expect this to be exercised but the framework allows for this

- Rendezvous tokens can be used as synchronization between parallel "threads"
  - RT purpose is to await each predecessor and aggregate results as needed
  - Details for inner workings ffs, e.g. maintain internal count for each invocation or backpointer to precursors
- Passing of results can be achieved using registers
  - Each token cell has an associated "input register"
    - Input register can be populated by predecessor T
    - Manifest token cell (parallel successors)
      - "Replicate" input across outputs
      - Each successor assigned separate input register
      - implies parallel registers (for "thread safety" – to avoid RACE conditions)
    - Rendezvous token cell
      - "Merge" inputs if needed
      - Merge function determined by rendezvous token type



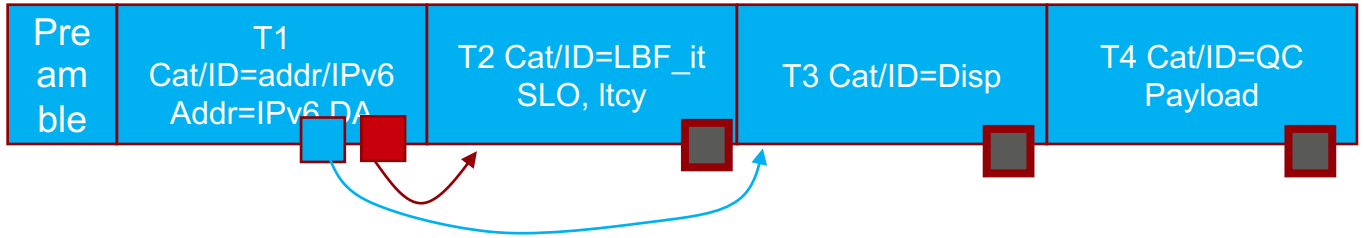


Disposition Token Cell

- Define what the network is to do with the packet when it leaves current TCR domain
- Analogous to:
  - Final next header in IPv6
  - MPLS bottom label
  - Network Programming IP suffix
- BUT without the constraints of size and with the advantage of larger, more flexible parameters

Example: Latency-Based Forwarding

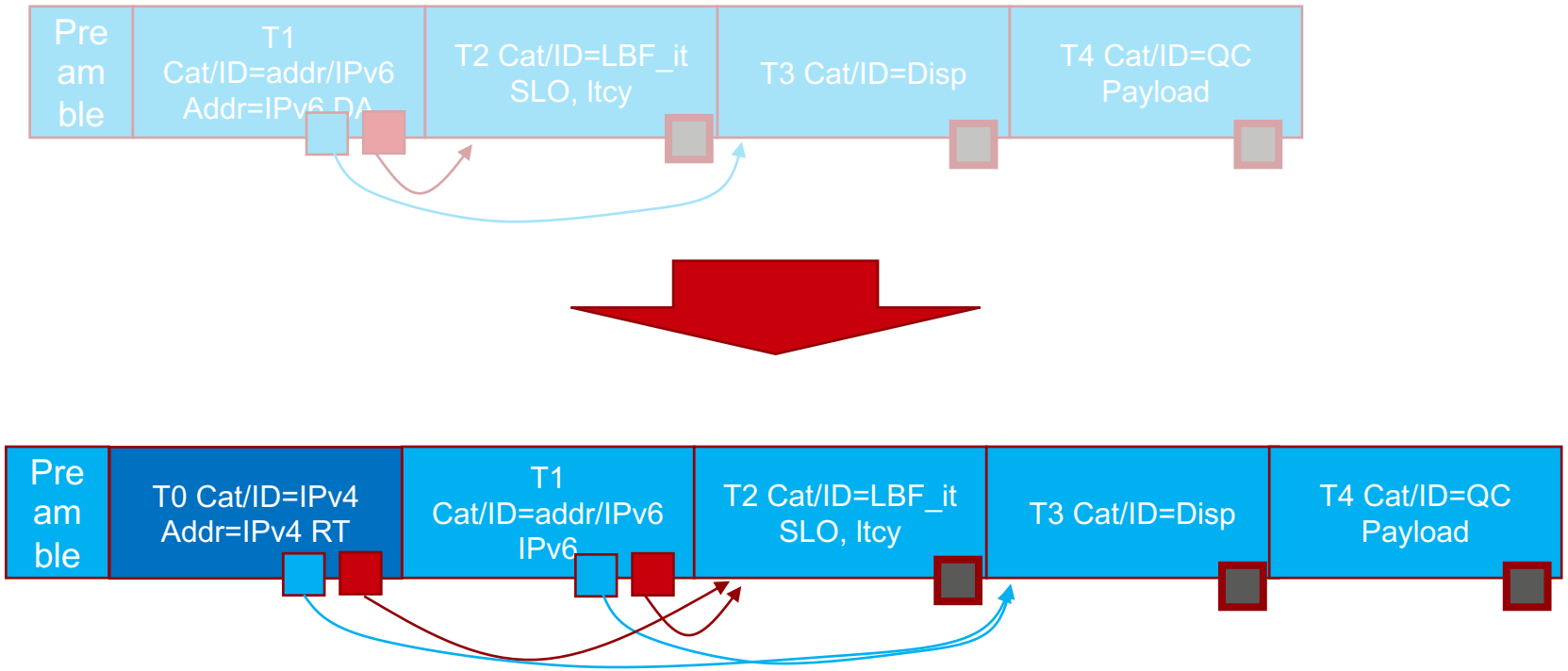
LBF specifies that a packet should undergo specific treatment to meet an SLO with a latency target



- *T1 results in forwarding decision and output interface*
  - *Red pointer: next token to process (target not reached)*
  - *Blue pointer: next token when target is reached*
- *T2 applies LBF algorithm to the output interface*

Example: FRR with Latency-Based Forwarding

FRR: add T0 for reroute target
LBF and end-to-end SLO are still applied!

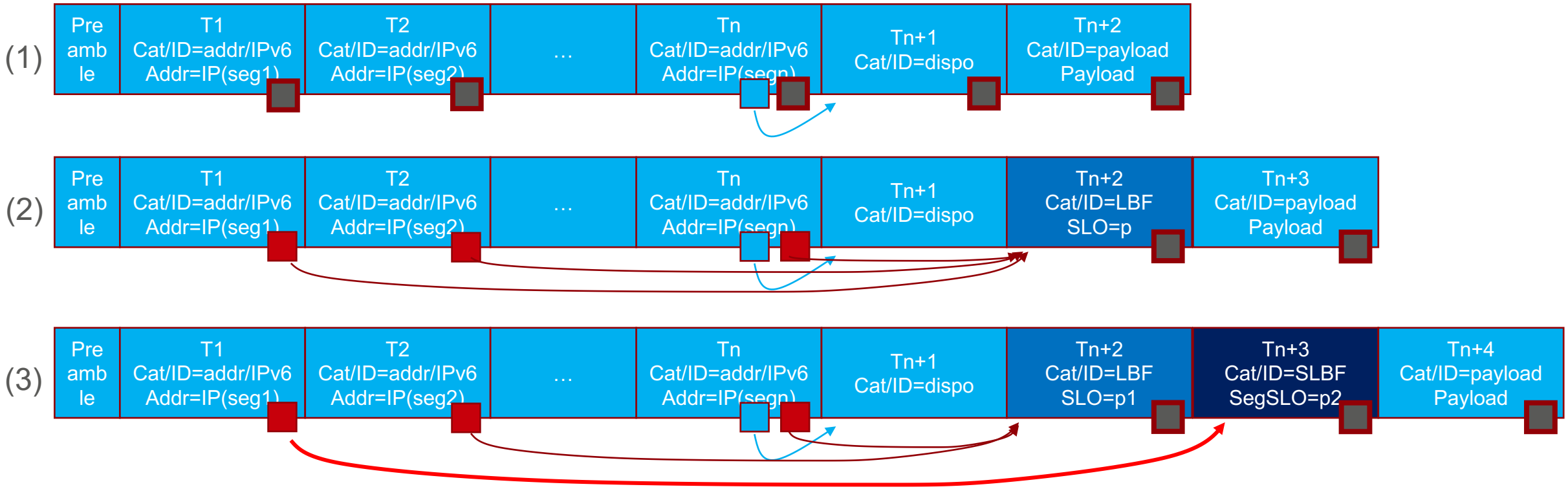


Example – Segment Routing ++

(1) Tokens represent segments

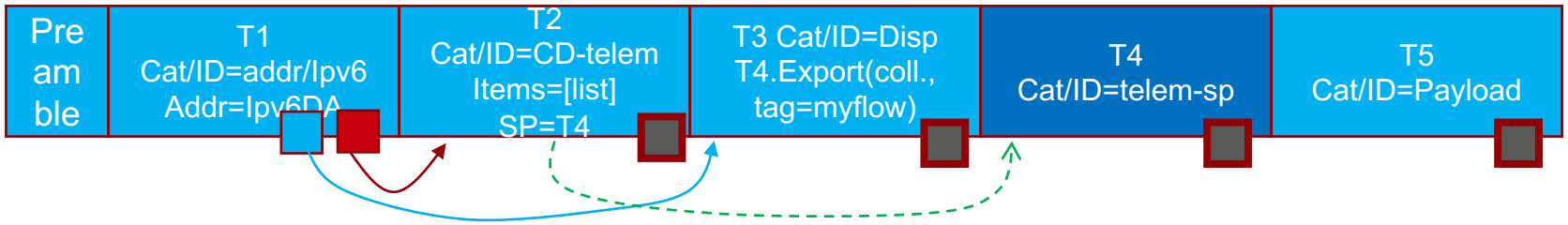
(2) Add LBF: add a token cell (pointed to as “next TC” by every segment TC)
LBF treatment is now given across all segments

(3) Perform differentiated SR with per-segment SLOs: add SegLBF TC per varying objective
This allows different SLOs and LBF treatment to be applied at different segments



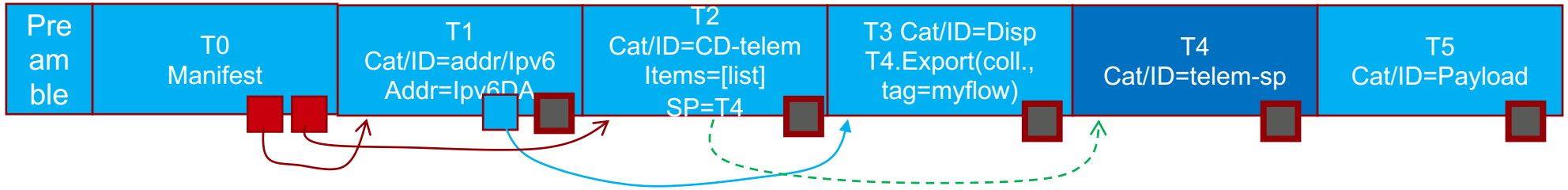
Example use case: telemetry & IOAM

Hop-by-hop along a path



T2 contains trace options: what to collect & where to record
*T4 is a **scratchpad***
writeable metadata that can be pointed to
scratchpad lifecycle is not tied to the tokens that write to it
disposition action includes how to dispose: eg export, log, discard

Optimized for concurrent processing / w/manifest

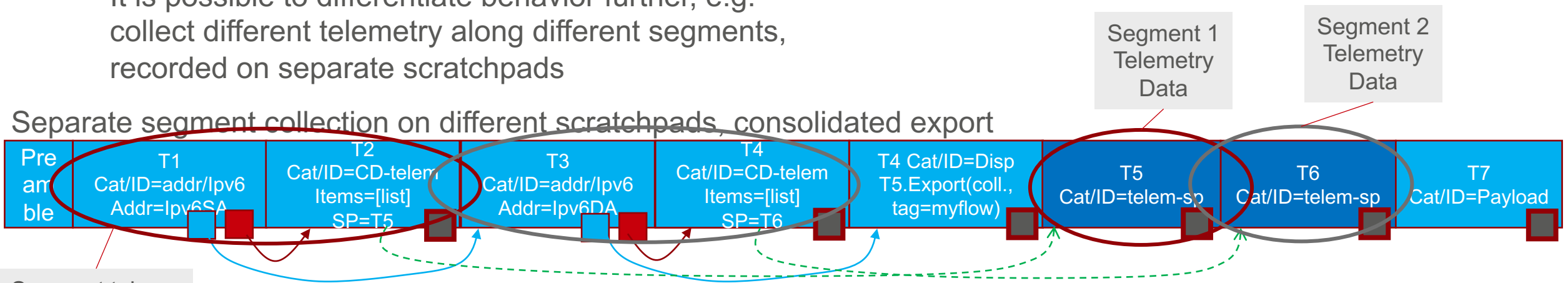


T0 specifies T1 and T2 can be executed concurrently

Example use case: telemetry & IOAM (contd.)

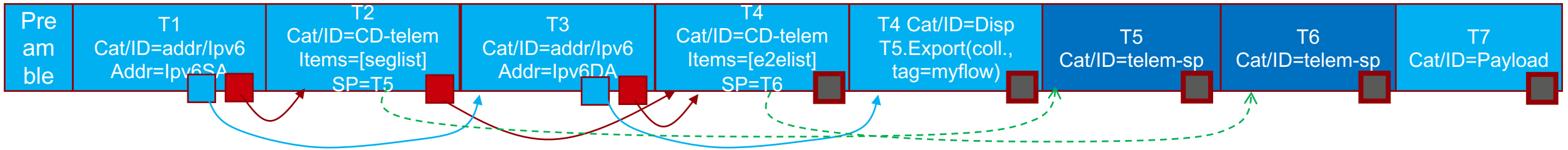
It is possible to differentiate behavior further, e.g. collect different telemetry along different segments, recorded on separate scratchpads

Separate segment collection on different scratchpads, consolidated export



Segment 1: T1 (forwarding), T2 (what to collect), T5 (where to record)
Segment 2: T3 (forwarding), T4 (what to collect), T6 (where to record)

End-to-end collection of set 1 plus segment collection of set 2 on different scratchpads

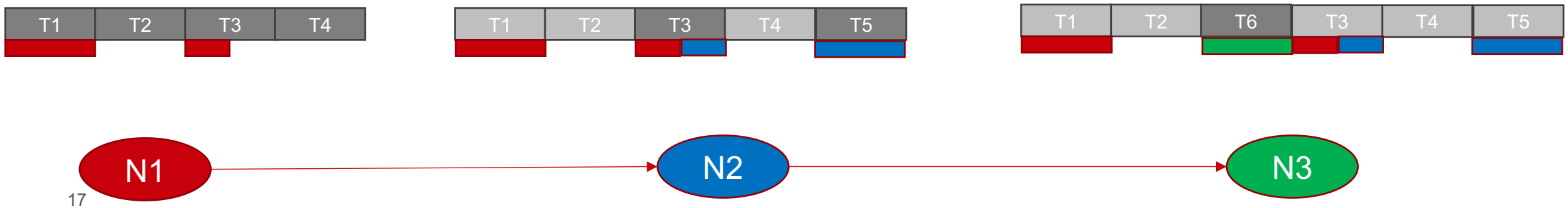


Complex behavior can be accomplished by combining different tokens into the same packet and linking them in different ways

Note, this could easily also be combined with LBF (one additional token to process)

Security and Authentication

- Token cells (and individual items within TCs) can be secured using Signature TCs
  - Identity of the signer
  - A “mask” of the material being signed
  - The signature itself
- Allow nodes to sign sets of TCs, and parts of TCs, without needing to sign the entire packet (or header)
- Allow multiple systems to sign different TCs in the same packet, and different portions of the same TC, allowing to verify authenticity of the contents regardless of who added it when manipulated across a path.



Conclusion

- TCR: A General-Purpose Networking Data Plane Protocol with Serializable and Parallelizable Token Cell Processing Flow
- Key differentiators
  - Serialization and Parallelization Framework
  - Scratchpads and Metadata in Token Cells
  - Extensible Token Cell Processing Engine
  - Differentiated Token Cell Security
- Multiple levels of programmability
  - Combine multiple token cells into a packet & link them in different ways
  - Token cell parameters (depending on type)
  - Introduction of new token cell types
- We would like to take this furher to explore the potential of the idea.
- If anyone is interested in working with us please reach out to either of the authors.

Thank you, Questions?