Train Communication Network

IEC 61375-2

Real Time Protocols

Message Services

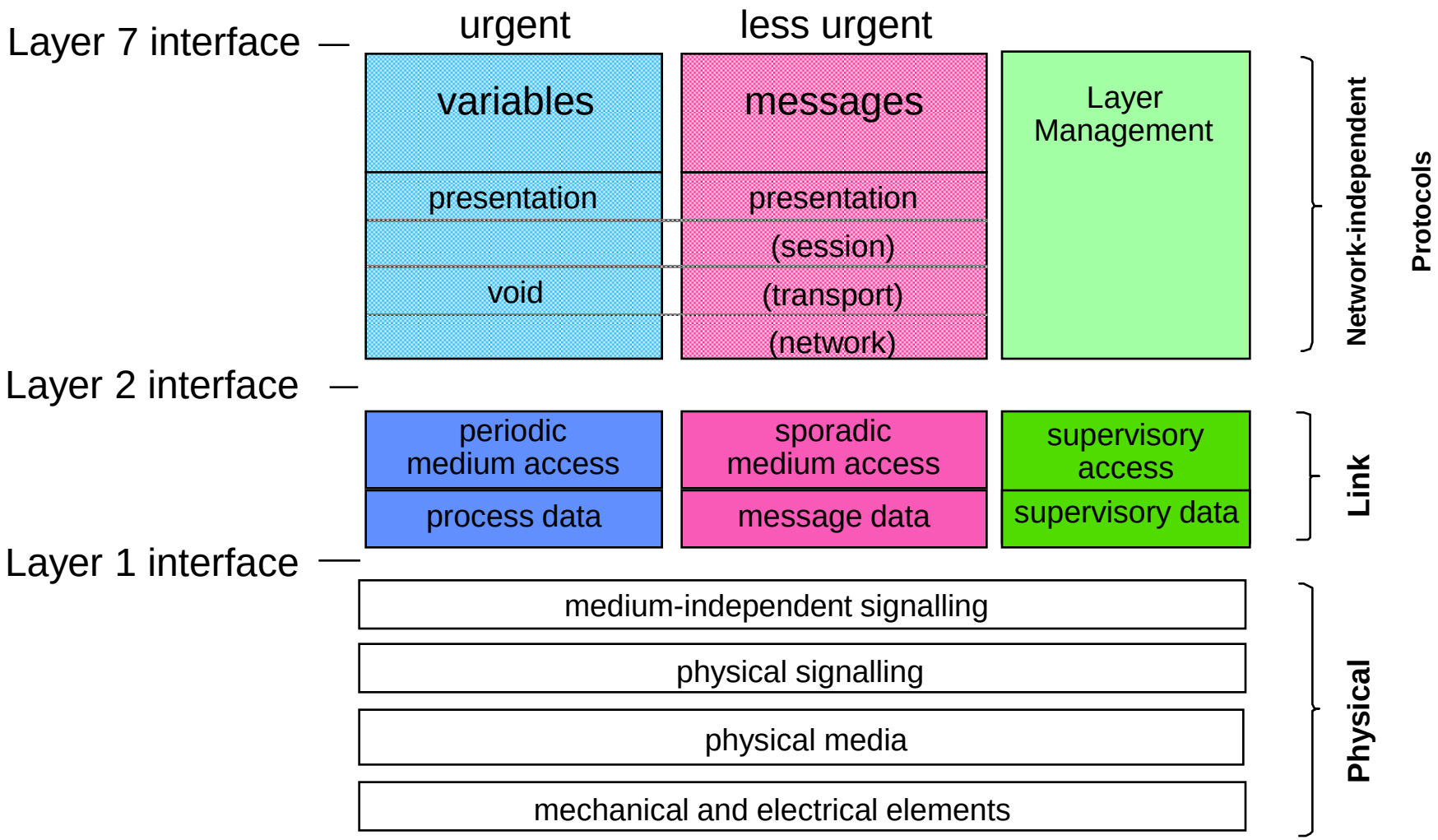
RTP- Message Services

1. General Principles

2. Variables
    1. Principle of cyclic Process Data broadcast
    2. Traffic Stores principle and implementation
    3. Process Variables and Datasets
    4. Software structure
    5. Application Layer Interface for Process Varialbles
    6. Networking

3. Messages
    1. Principle of Message Data communication
    2. Link Layer Interface
    3. Networking and Routing
    4. Transport protocol
    5. Software structure
    6. Application Interface

TCN Stack


urgent
less urgent
Layer 7 interface
variables
messages
Layer Management
presentation
presentation
(session)
void
(transport)
(network)
Network-independent
Protocols
Layer 2 interface
periodic medium access
sporadic medium access
supervisory access
process data
message data
supervisory data
Link
Layer 1 interface
medium-independent signalling
physical signalling
physical media
mechanical and electrical elements
Physical

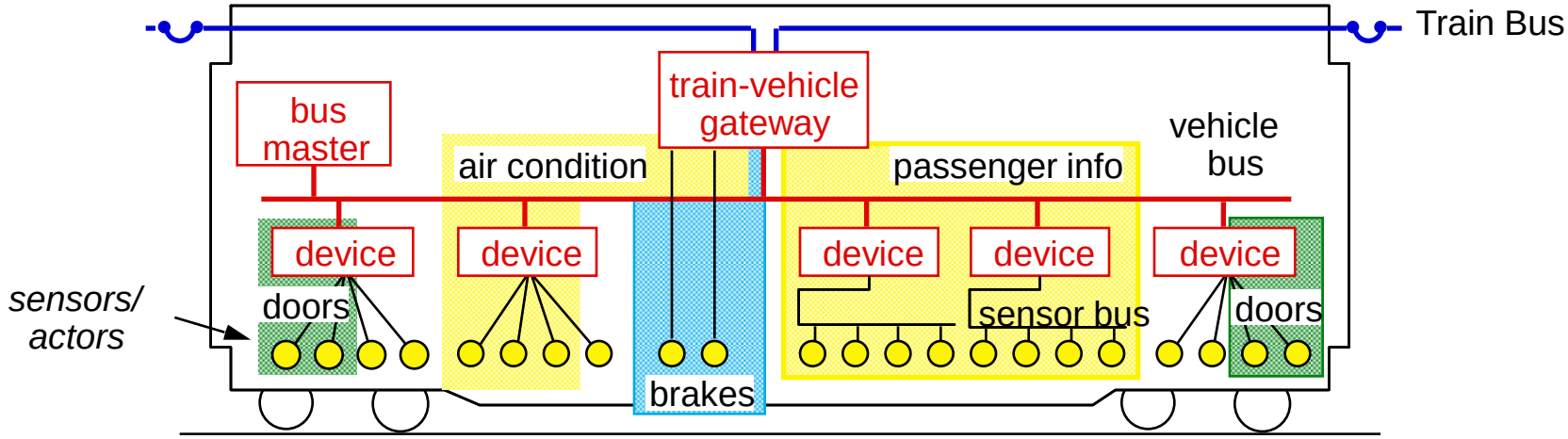
Functions and Devices

Each vehicle supports a number of standardized functions.

The Train Bus accesses vehicles without knowing their internal structure.

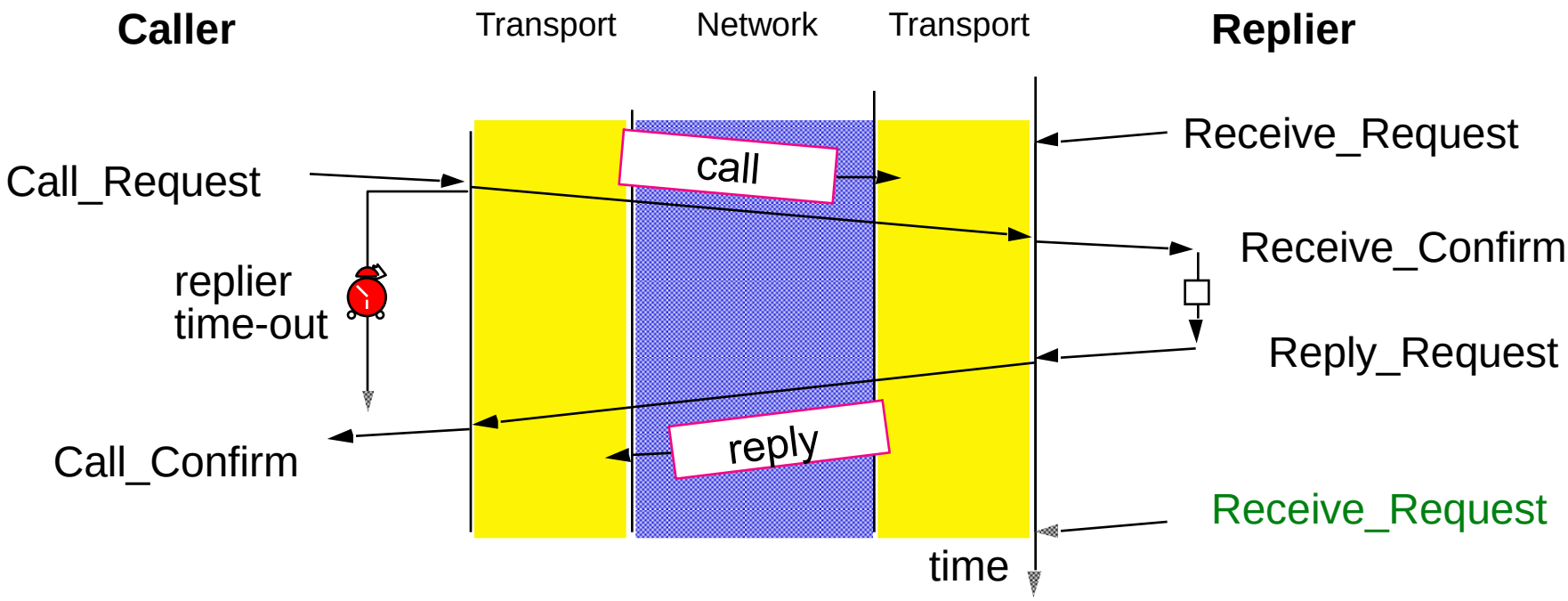
The train bus accesses *functions* rather than *devices*.

These functions are implemented by one or several vehicle bus devices, or even by the gateway itself.



The gateway deduces the device from the function and routes messages.

Client-Server Service



The Application Interface for Messages provides a "Call with Reply" service

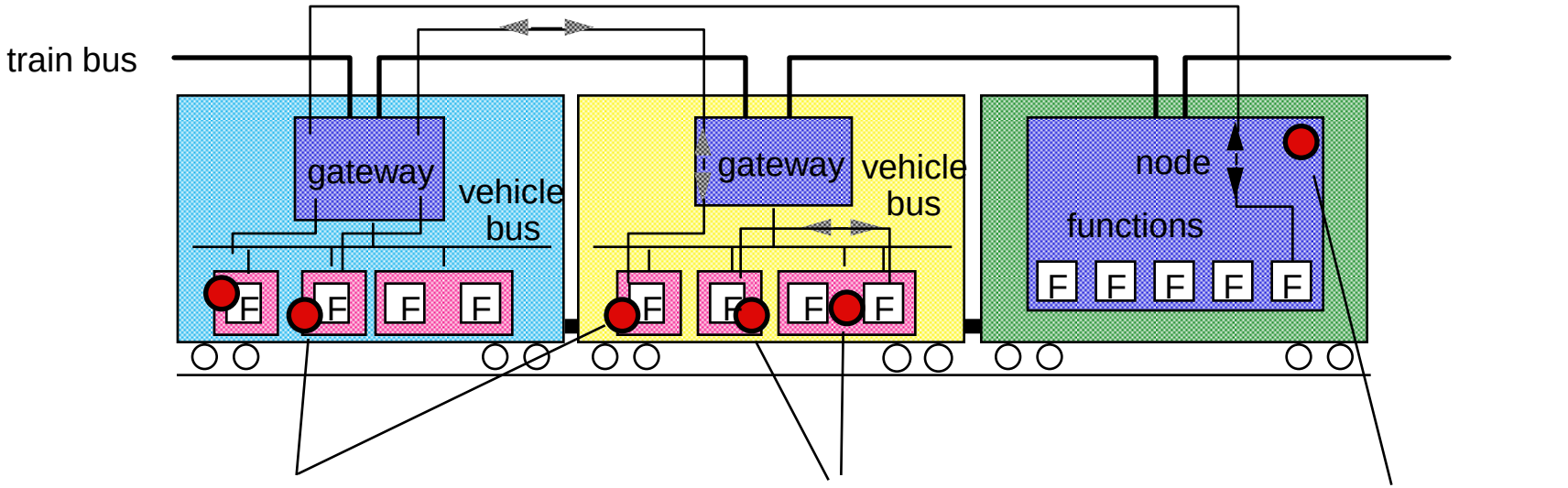
Applications communicate among themselves on a Client/Server basis.

Tasks use the same communication scheme:

- within the same processor
- within the same vehicle bus and
- within the Train Communication Network

Transport Protocol for Messages

The transport protocol ensures a reliable communication from end-to-end between *Application Functions.*



the message transport protocol is executed by the vehicle bus devices

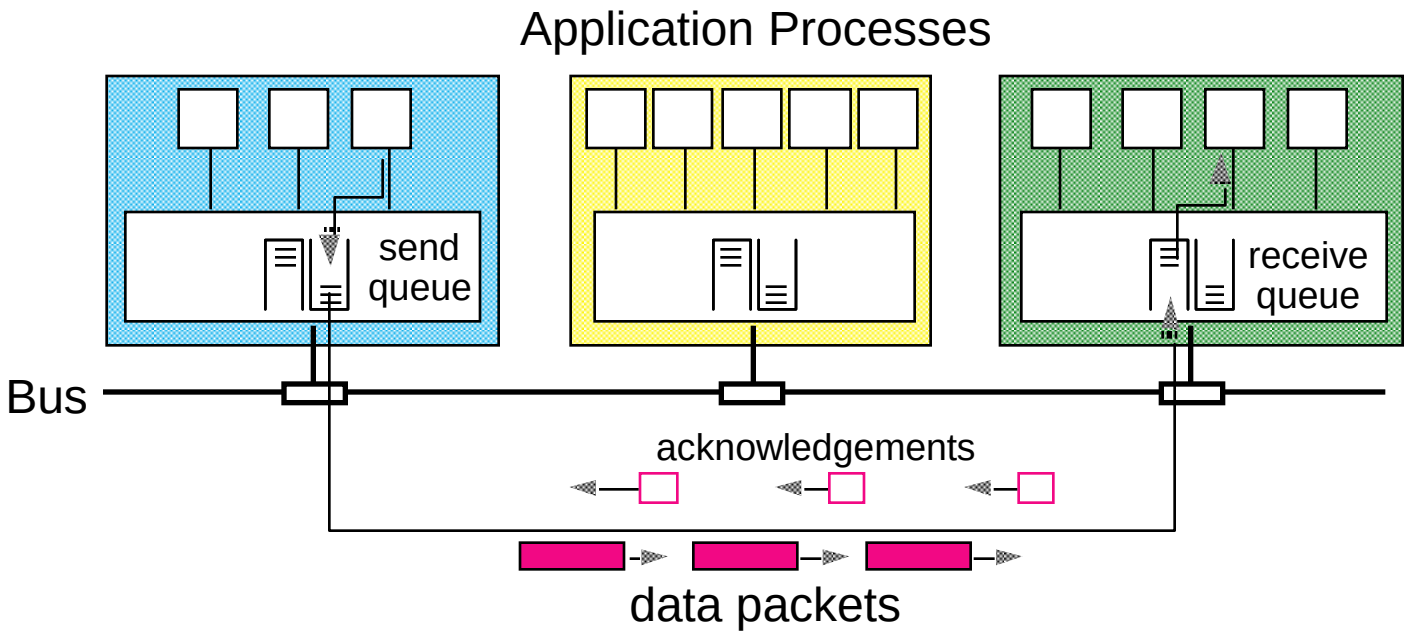
the message transport protocol applies also within the same bus and within the same device.

in vehicles without vehicle bus, the message transport protocol is executed by the train bus node

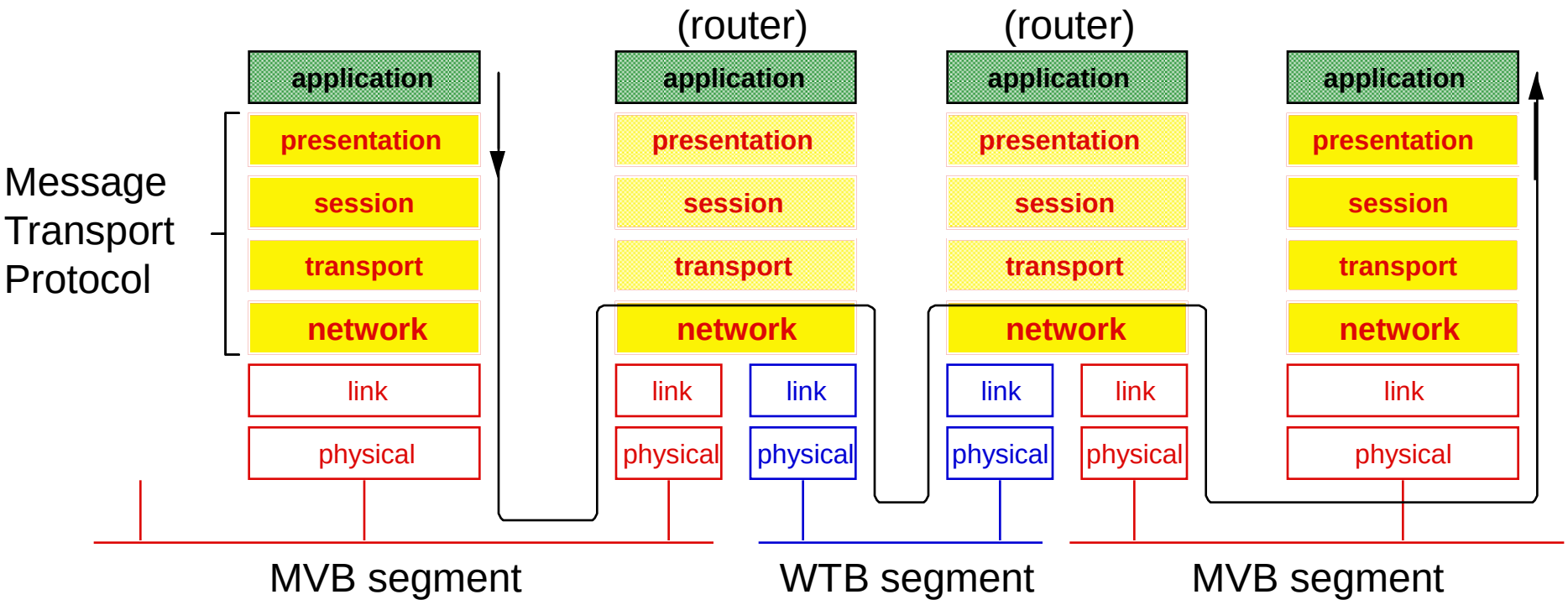
The Message Transport Protocol runs in each device

Message Data Transmission

Messages are lengthy, but not so urgent data.
They are used e.g. for diagnostics, passenger information, down-loading.
Messages are segmented into *packets* for transmission.
Data, ackowledgements and control packets form the *Message Data*.
Message Data are sent upon demand between two process data cycles.
The sender and receivers of Message Data are queues (no buffers):



End-to-end Transport Protocol



Porting the MTP to a bus providing connectionless datagrams is easy

RTP- Network Layer

1. General Principles

2. Process Data
    1. Principle of cyclic Process Data broadcast
    2. Traffic Stores principle and implementation
    3. Process Variables and Datasets
    4. Software structure
    5. Application Layer Interface for Process Variables
    6. Networking

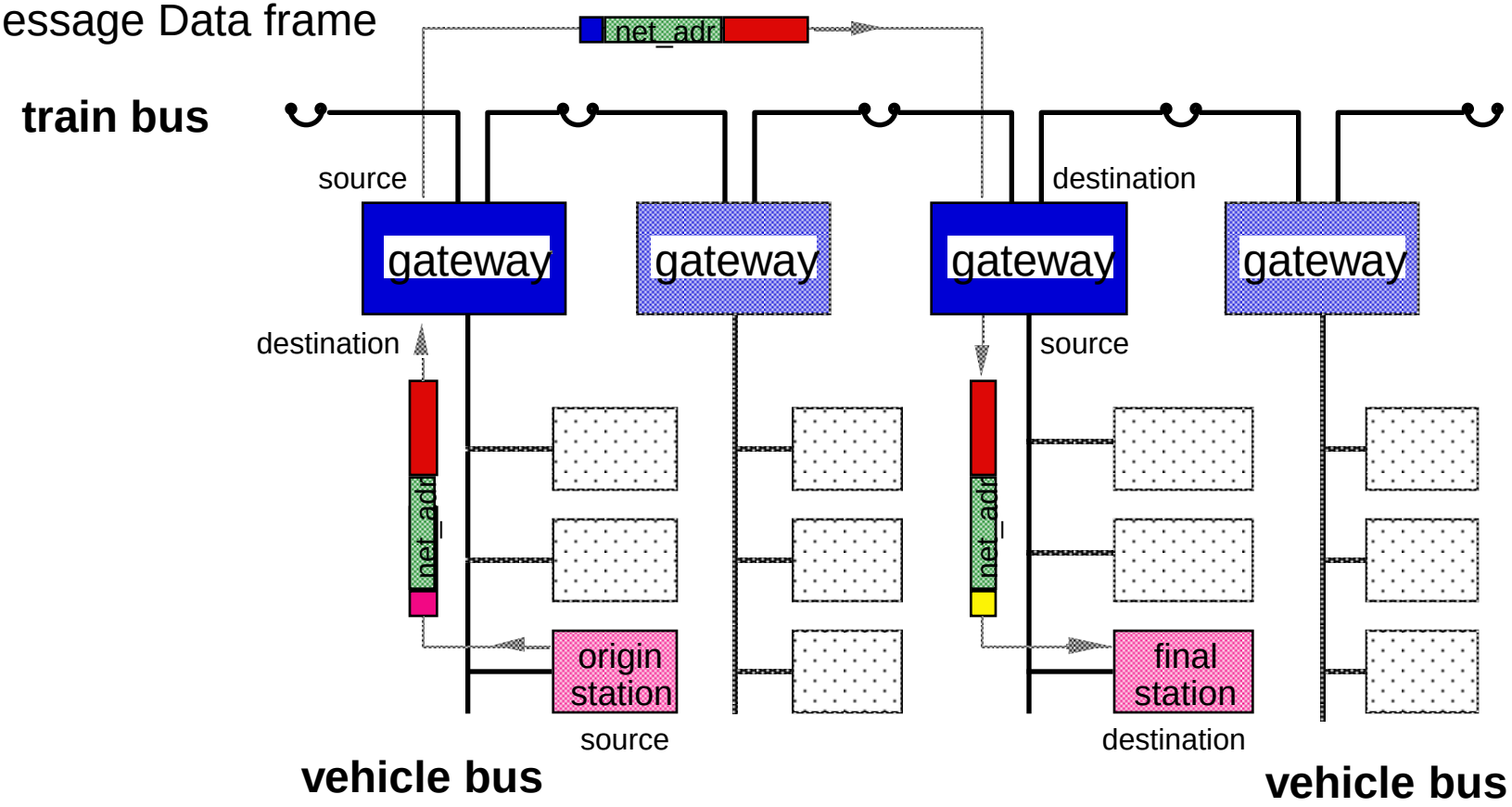
3. Messages
    1. Principle of Messages communication
    2. Link Layer Interface
    3. Networking and Routing
    4. Transport and Session protocol
    5. Software structure
    6. Application Interface

Network: Packet Routing

The gateway operates as a*router* : it forwards packets from bus to bus without keeping knowledge of previous packets.

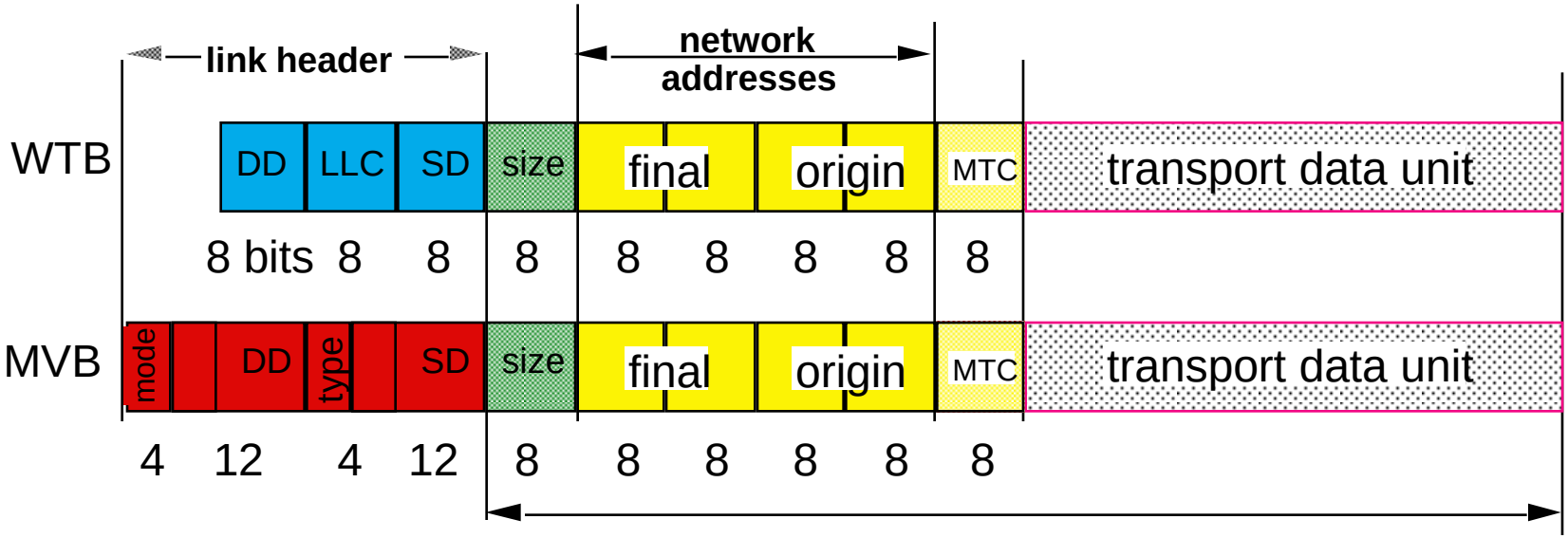
Packets are transported in Message Data frames as *datagrams* which contain the full origin and final address.

Message Data frame



Network: Message Data Frames

Message Data carry data packets, acknowledgements and control data.

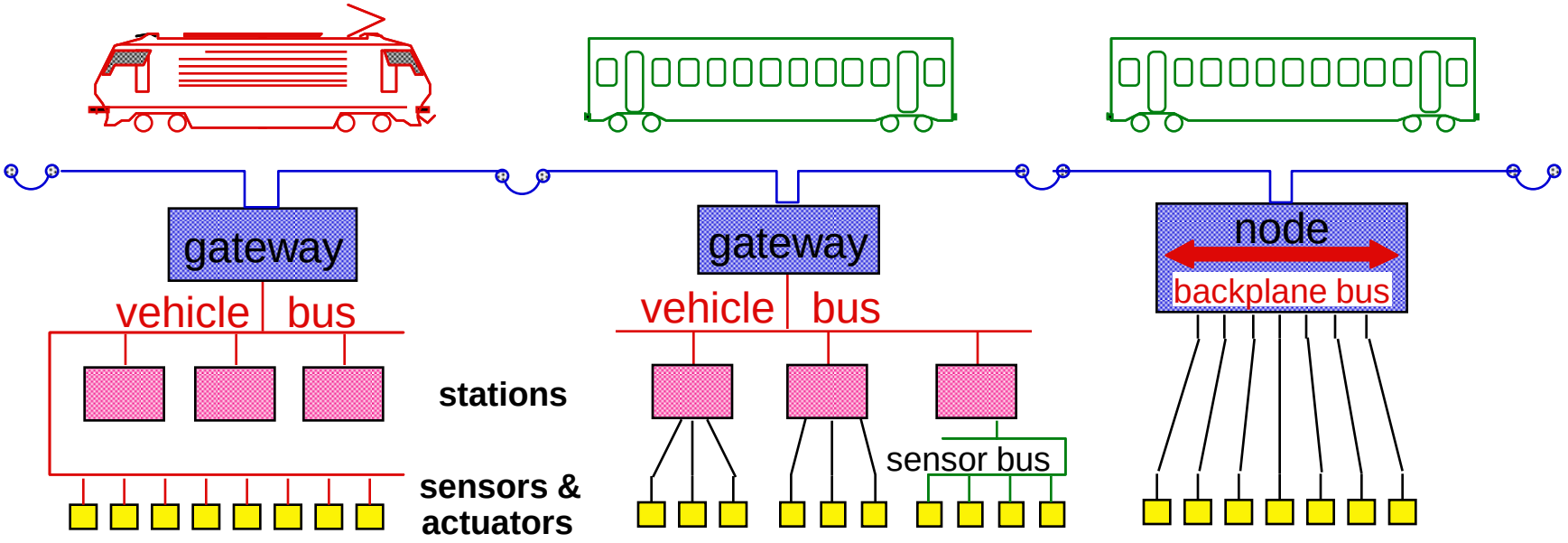


DD: destination device
SD: source device
LLC: link layer control
MTC: Messsage Transport Control

Message Data have the same format on the vehicle or on the train bus.

They are datagrams, which carry the full origin and final address

Networking Different Busses



2-level hierarchy
Intelligent stations and sensors/actuators are attached to the vehicle bus

3-level hierachy
Sensors are attached directly or by a sensor bus

1-level hierarchy
Equipments are attached to a node-internal backplane bus which plays the role of a vehicle bus

The real-time protocols allow to interconnect different vehicle structures

Network Layer Operation

The network layer is responsible for the routing of packets (data, acknowledgement and control) through the network

Routing relies on the network addresses contained in each packet.

It is connectionless, i.e. it retains no knowledge about previous packets belonging to the same message

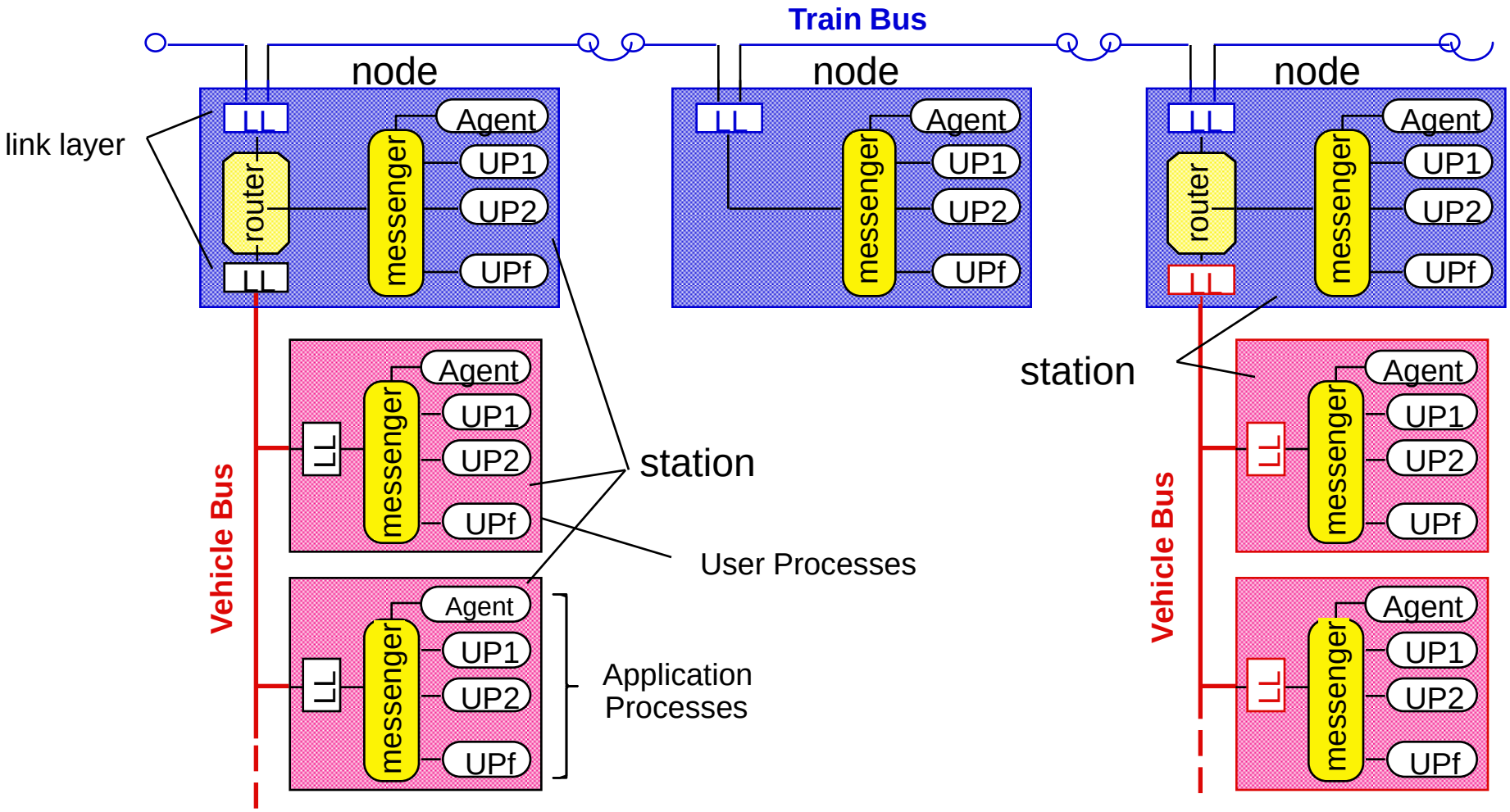
Routing is done on the base of two directory tables:

- station directory
- function directory

These directories are set up by the application or by network management.

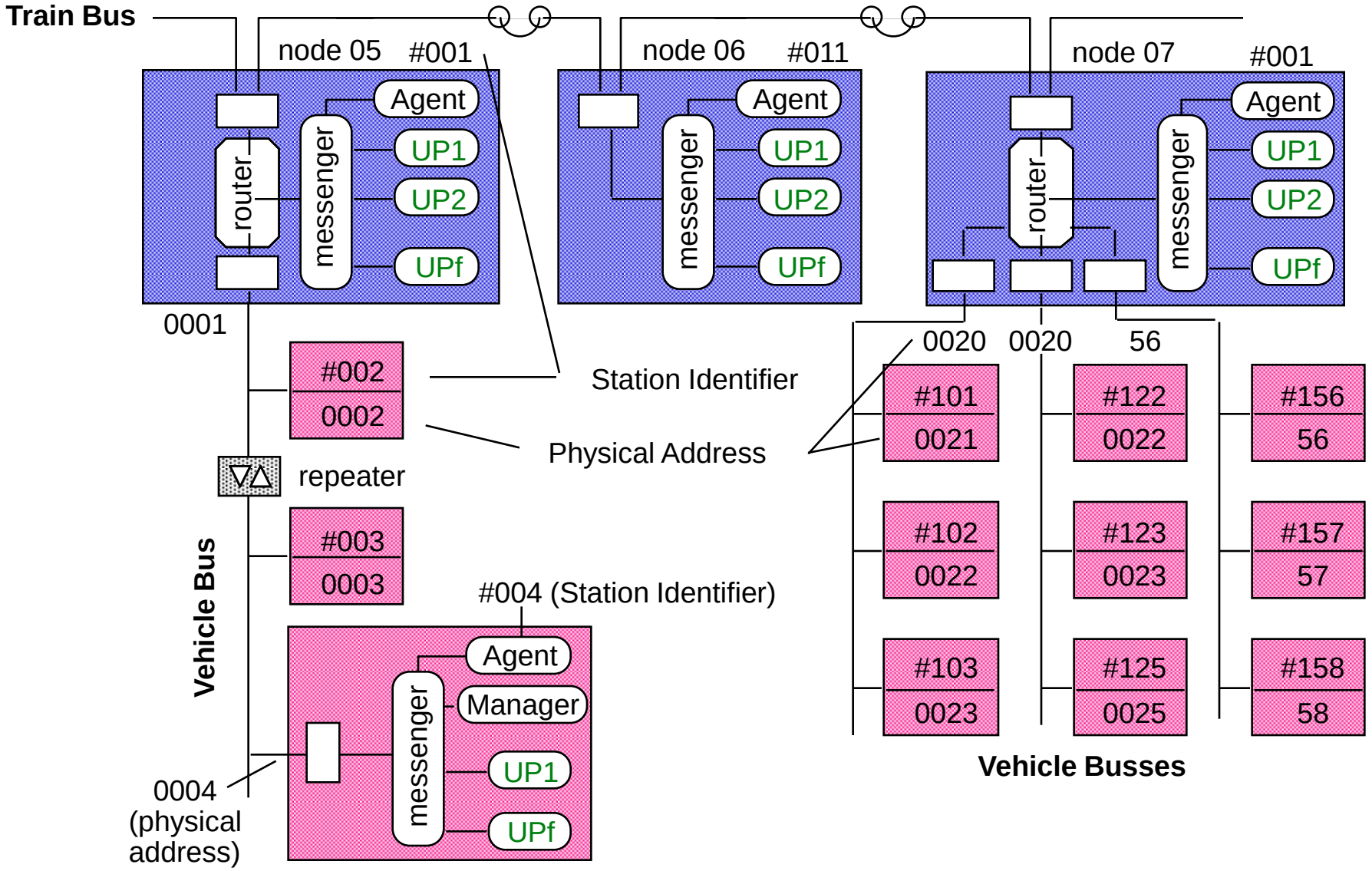
The network layer has no protocols (no segmentation / reassembly), but address calculation is complex.

Network: Reference Architecture

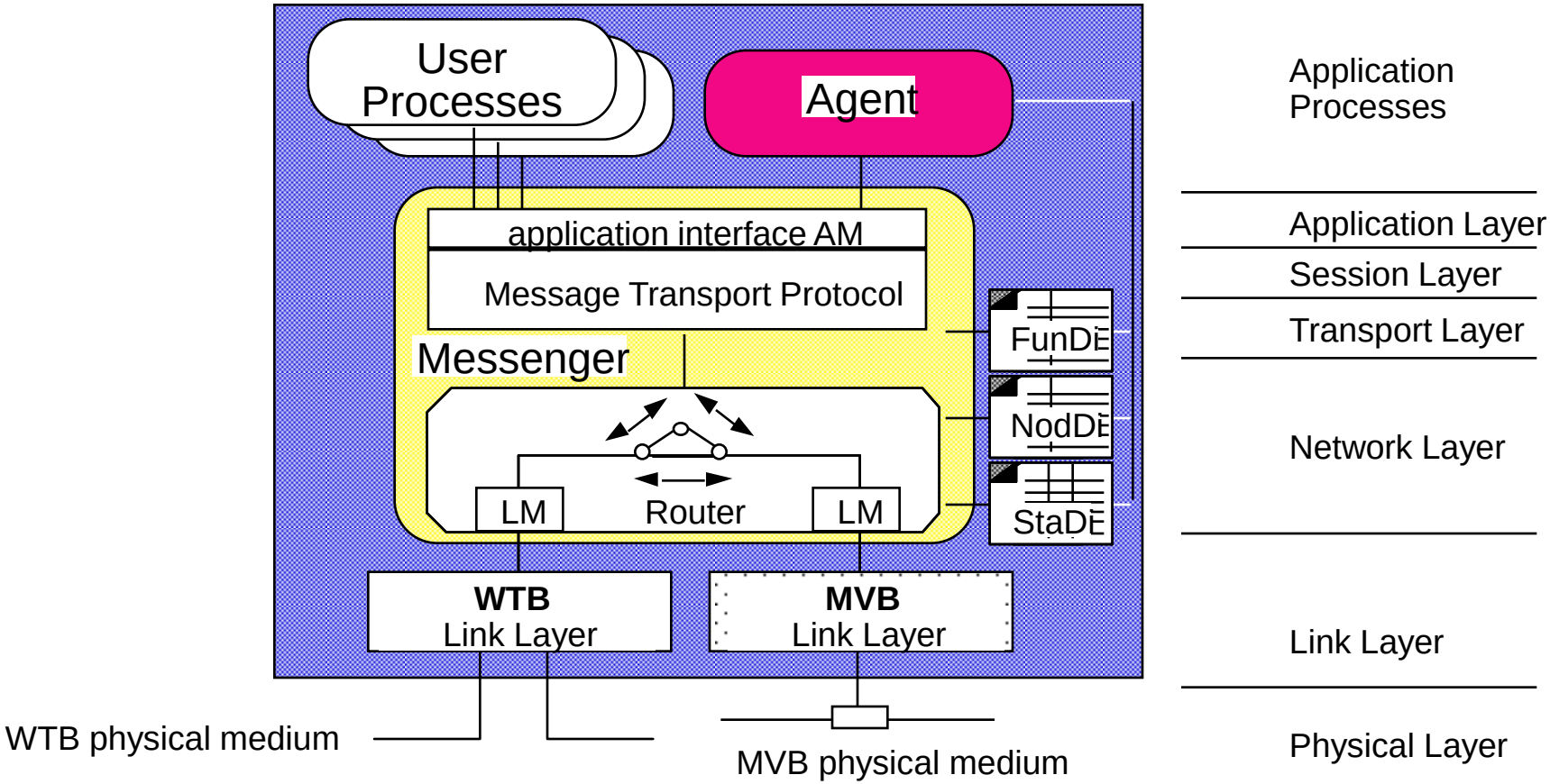


All Application Processes (UPs, Agents) communicate through the Messenger.
A station is a (vehicle or train bus) device capable of message communication.

Network: Example of an Actual Configuration



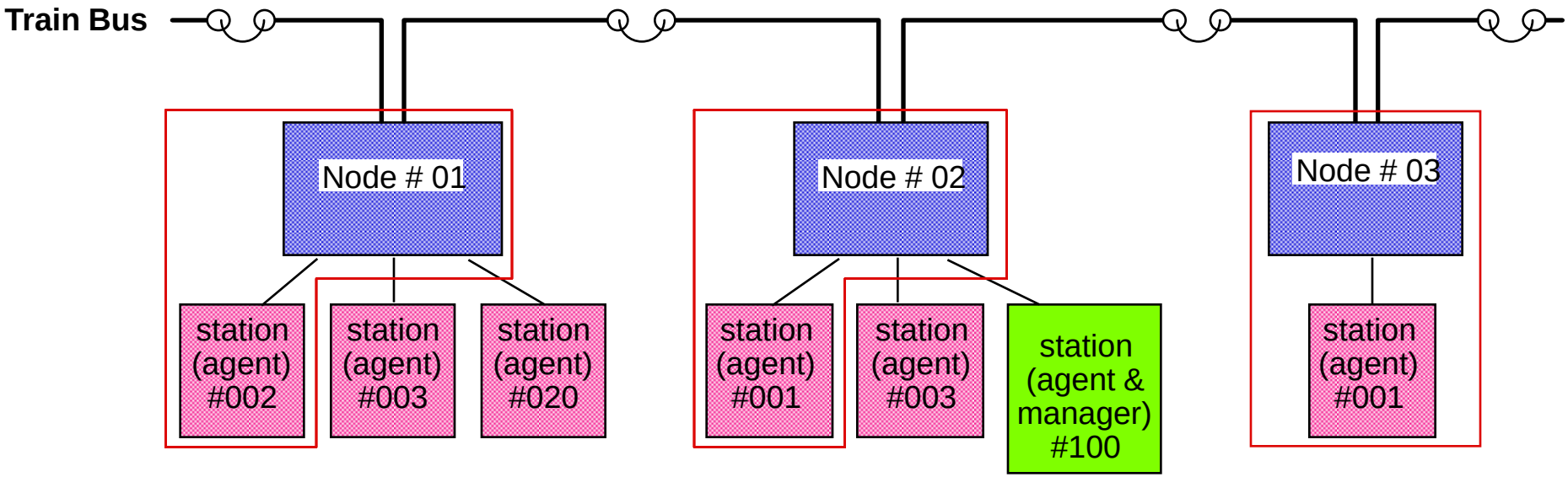
Network: Reference Gateway



Agent = Station Management Agent
(**for parametrizing, down-loading, configuration, debugging, performance measurement)**

A gateway has a router and more than one link layer

Network: System Point Of View



The System Engineer identifies stations attached to train bus nodes.

The train bus node counts as one station.

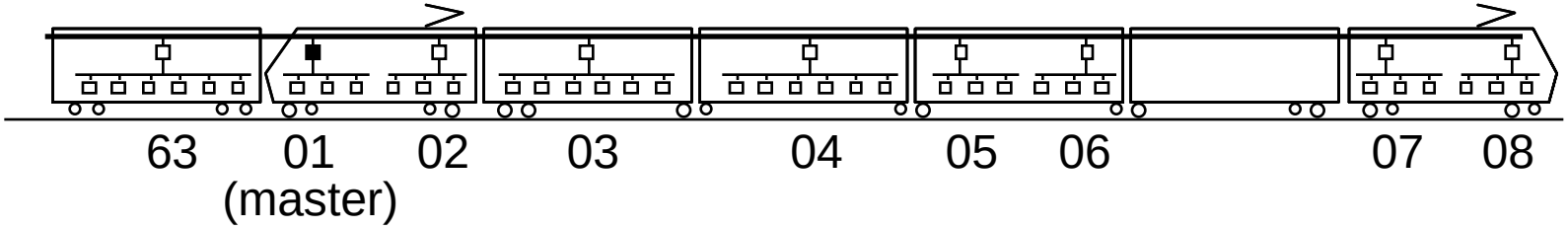
The interconnection of the stations is not visible (no, 1, 2,... vehicle busses).

The communicating entities are the Agents and the Managers.

The nodes route the packets through their Station Directory

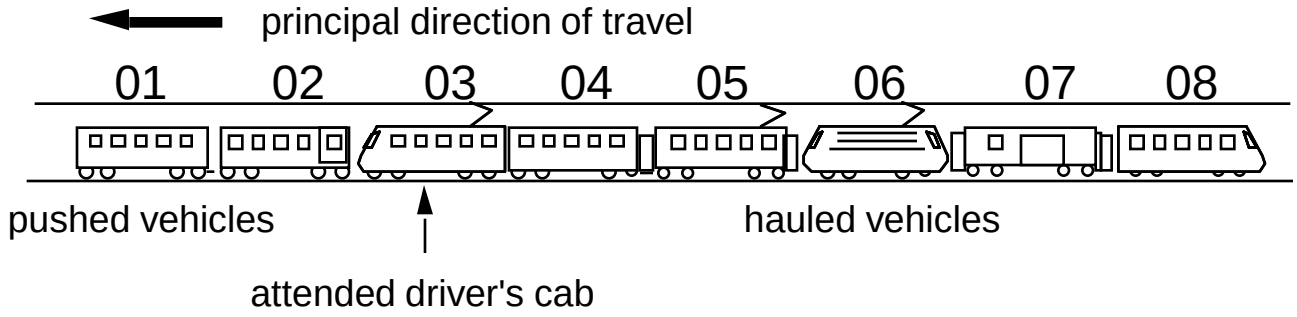
Network: Train Bus Addresses

The Train Bus operates with nodes addresses, which is the position of a node relative to the master node (address 01)



The *train inauguration* gives each node its position and the direction of the master.

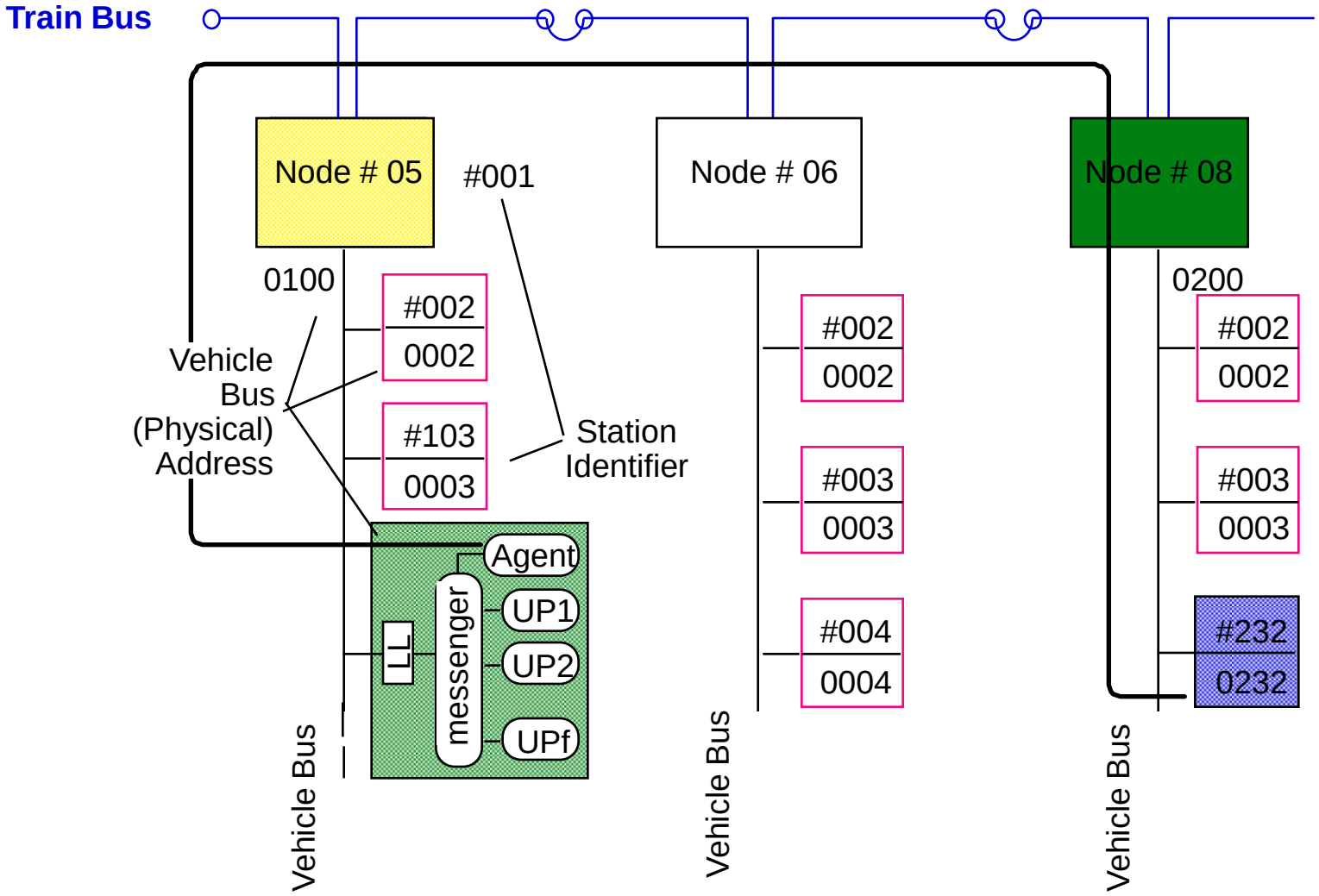
UIC applications address *vehicles* by their position relative to the head of the train.



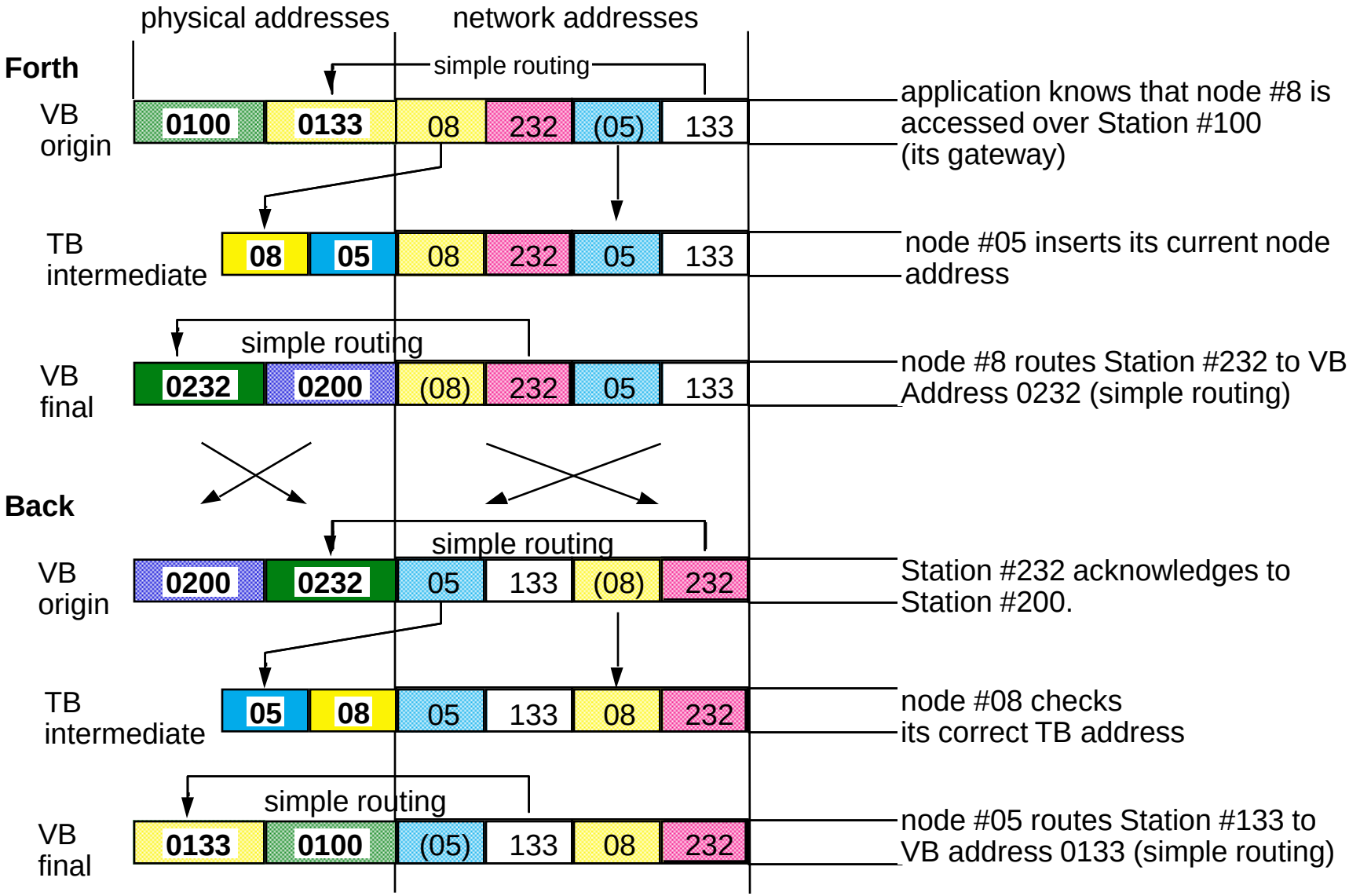
Since a vehicle may have one, two or no operative node, there must exist a *mapping* between node address and UIC application address.

The real-time protocols only rely on the TCN addressing.

Network: Example Of Communication



Network: Example of Frame Exchange

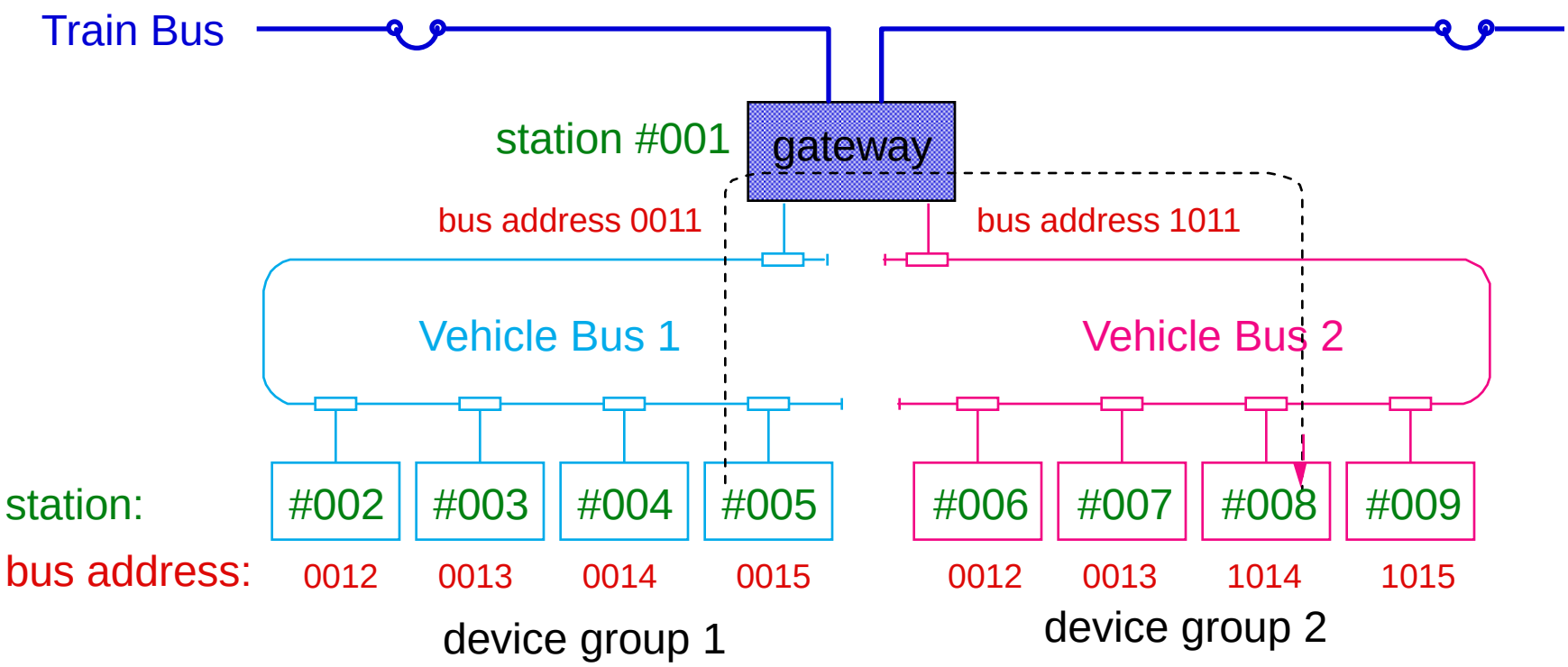


Network: Station Directory

The Station Directory in the network layer routes the messages on the base of their station address to the corresponding link layer and device address

Station	Link Layer	Physical Address
003	MVB1	0003
005	MVB1	0005
103	MVB2	0003
223	Parallel_Bus	203040

Network: Routing Between Two Vehicle Busses

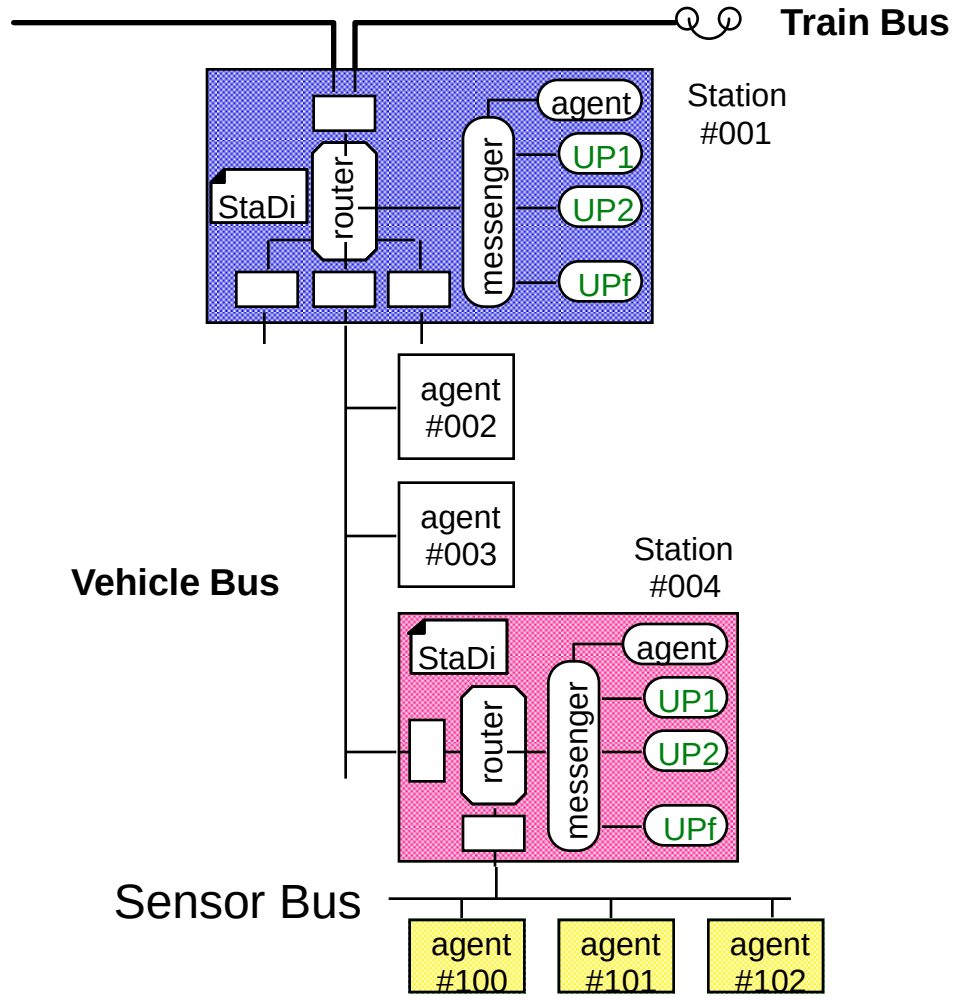


Two (or more) vehicle busses may be attached to a train bus gateway.

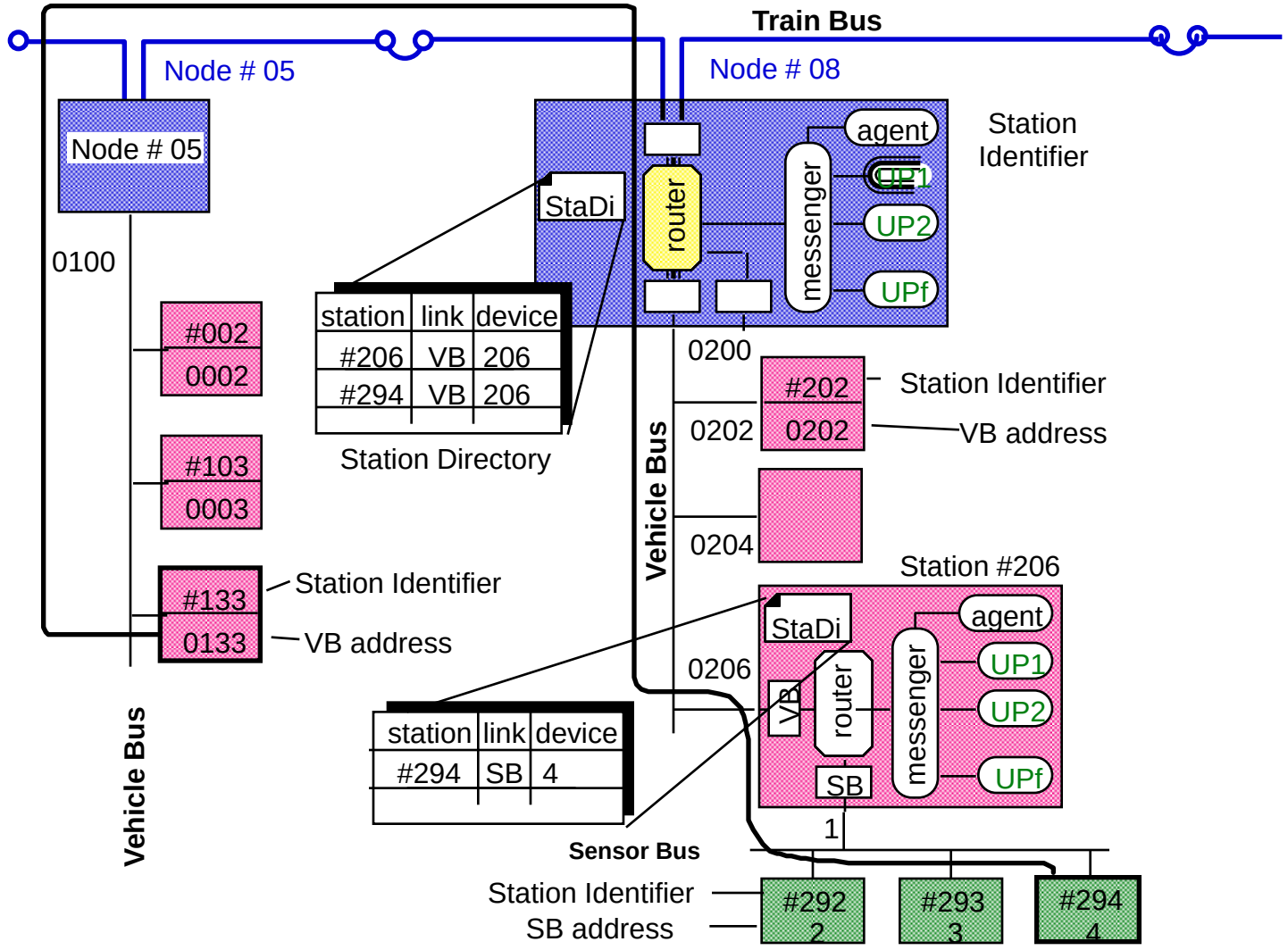
The gateway routes message data according to the network address from vehicle bus to vehicle bus or from vehicle bus to train bus.

The station identifier must be unique under a given train bus node.

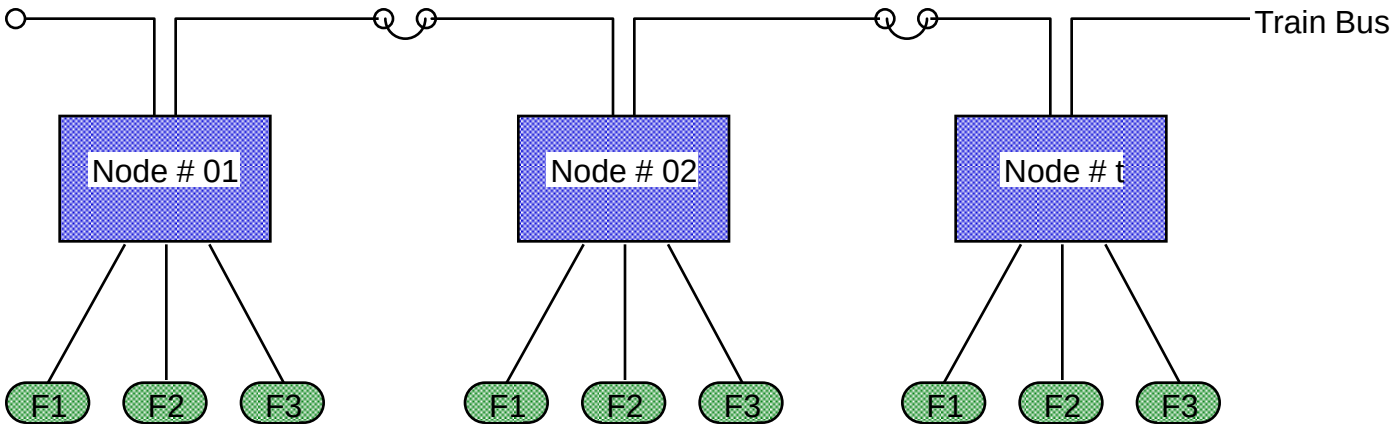
Network: Sensor Bus Configuration



Network: Sensor Bus Path



Network: User Point Of View

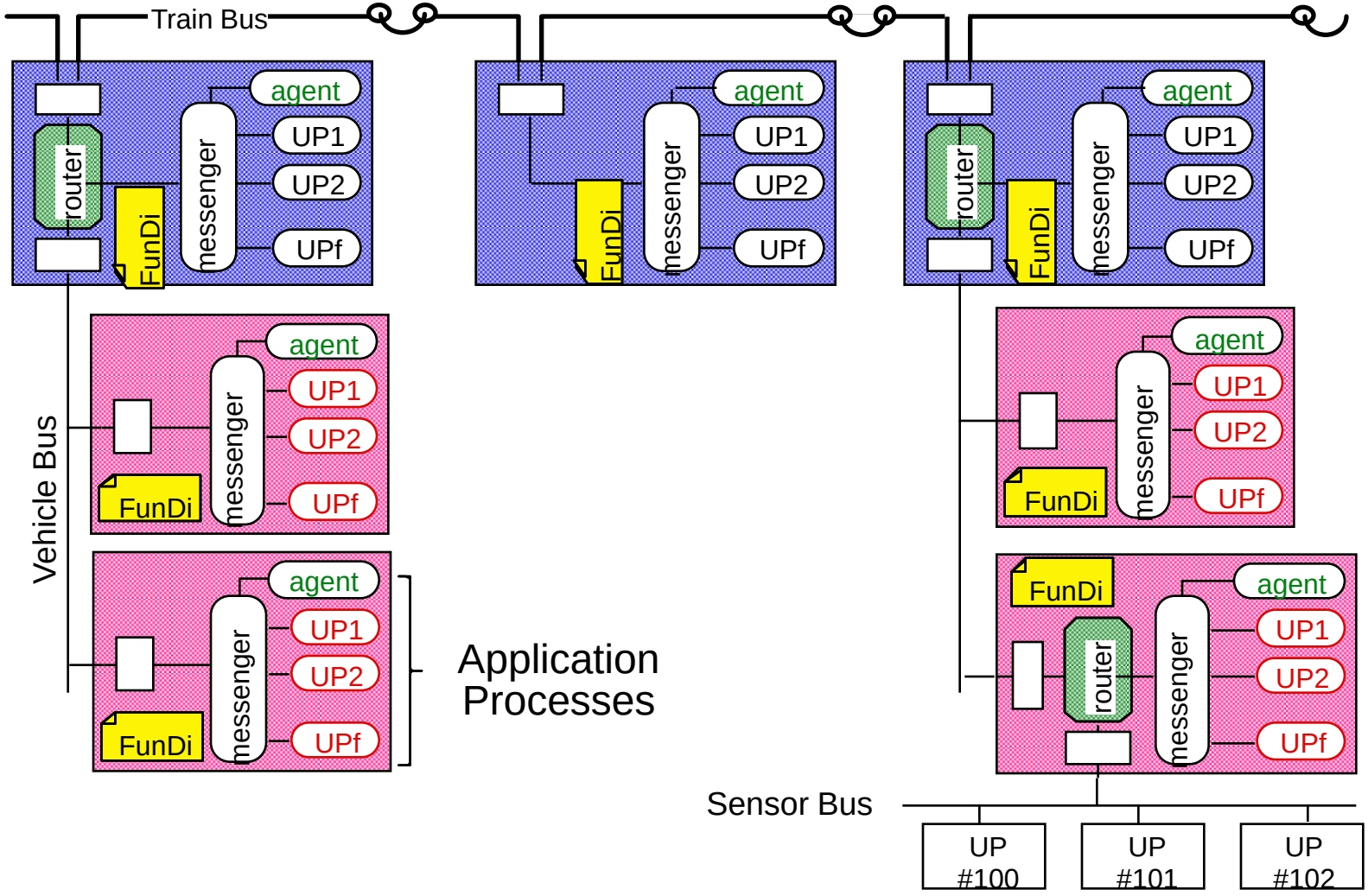


The user sees functions attached to nodes.

The node itself is not a function - but the node device can implement functions

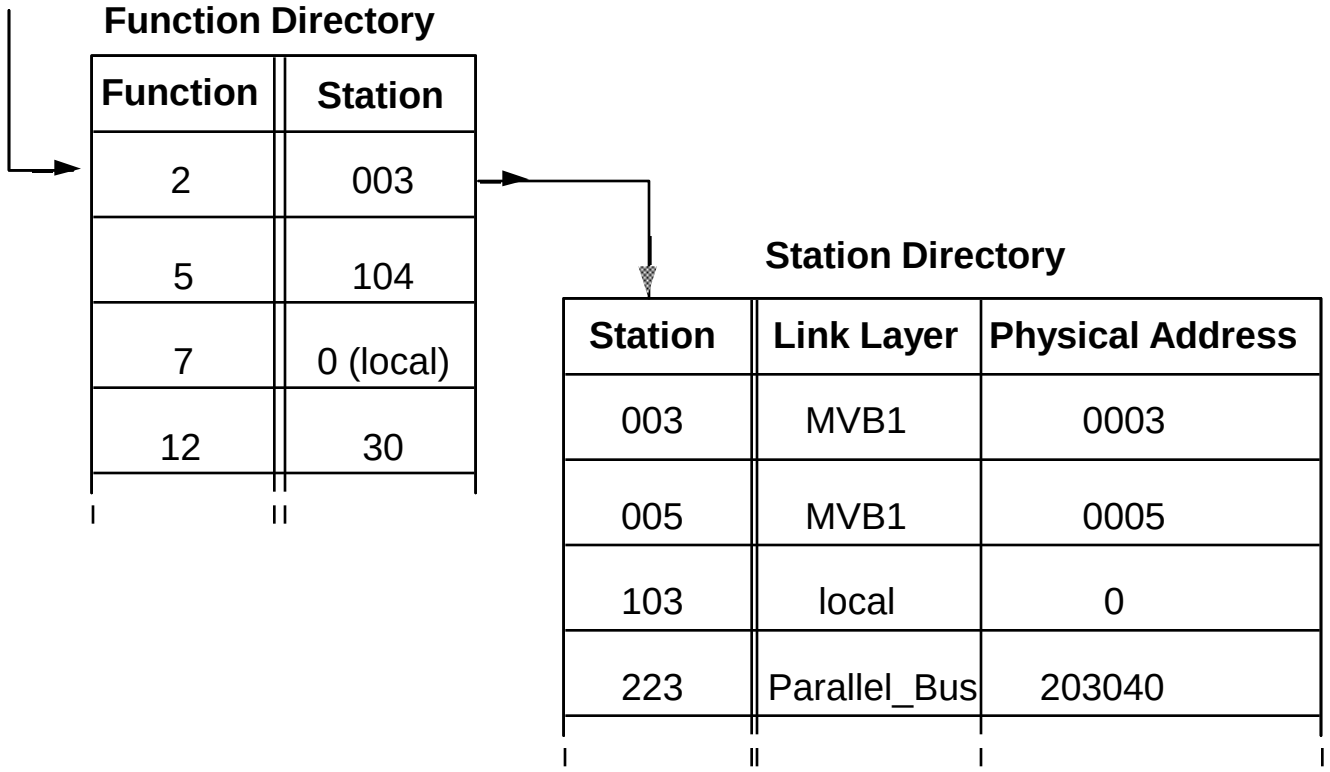
Network: Function Architecture

The communicating entities are functions in the different stations



Network: Function Directory

The Function Directory maps the function to the corresponding station



Network: Address Kinds

Physical Address:
identifies a device on a bus - can be a broadcast address.
gateways have more than one physical address
each frame carries the source and destination address.

Link Identifier:
In a gateway, link layers are identified by their address (e.g. 1,2)

Link Service Access Point
Only one default LSAP is used in TCN (unused).

Network Address:
The application process identifies the remote application process through its network address.

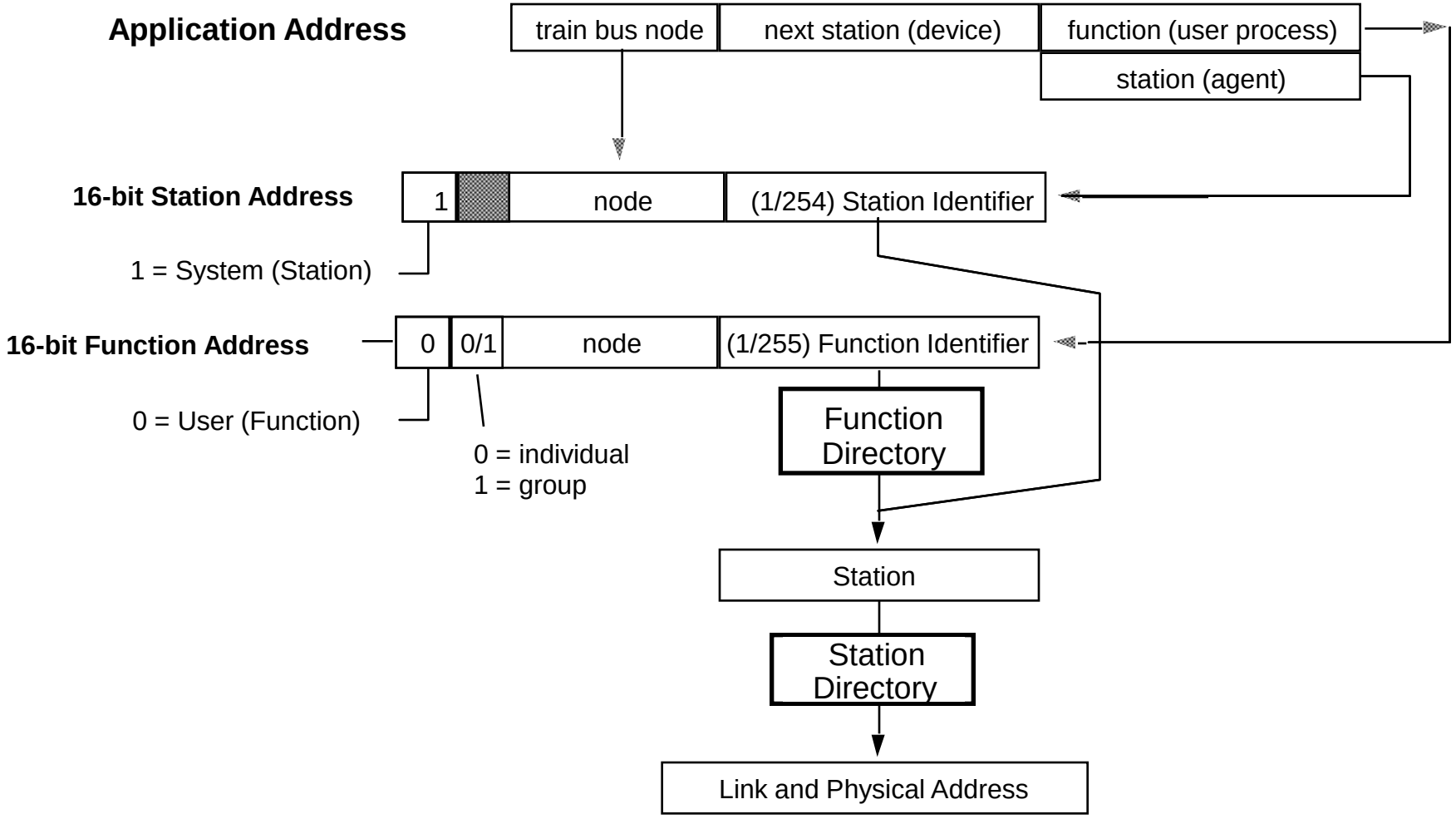
Each frame carries the *origin* and *final* network address

The network address is the concadenation node + (station or function)

Application Address:
All other addresses are deduced from this one.

system/user	1 bit
individual/group	1 bit
node	6 bits
station or function	8 bits
next station	8 bits

Network: Address Calculation



Network: Message Consistency and WTB Topography

Transport level:

if a topography change occurs while a message is transmitted, the remaining packets may be delivered to the wrong node.

Session level:

if a topography change occurs between a call and a reply message, the reply message may be delivered to the wrong node.

This misaddressing will be detected in most cases.

When nodes swap their addresses, undetectable situations may occur.

When topography changes, each WTB node signals its messenger, which cancels all ongoing message data communication over the WTB and flushes the queues of the WTB link layer.

To this effect, each WTB node maintains a topography counter, which is incremented each time the topography of the WTB changes.

All packets of a message exchanged over the vehicle bus carry the topography counter in place of the local node address (which is redundant).
If a station detects that topography changed, it cancels only that conversation.

Network: Impact of Topography Change on Communication

A topography change cancels indistinctly all conversations over the WTB.

It would not be necessary to cancel conversations between nodes which did not change their addresses.

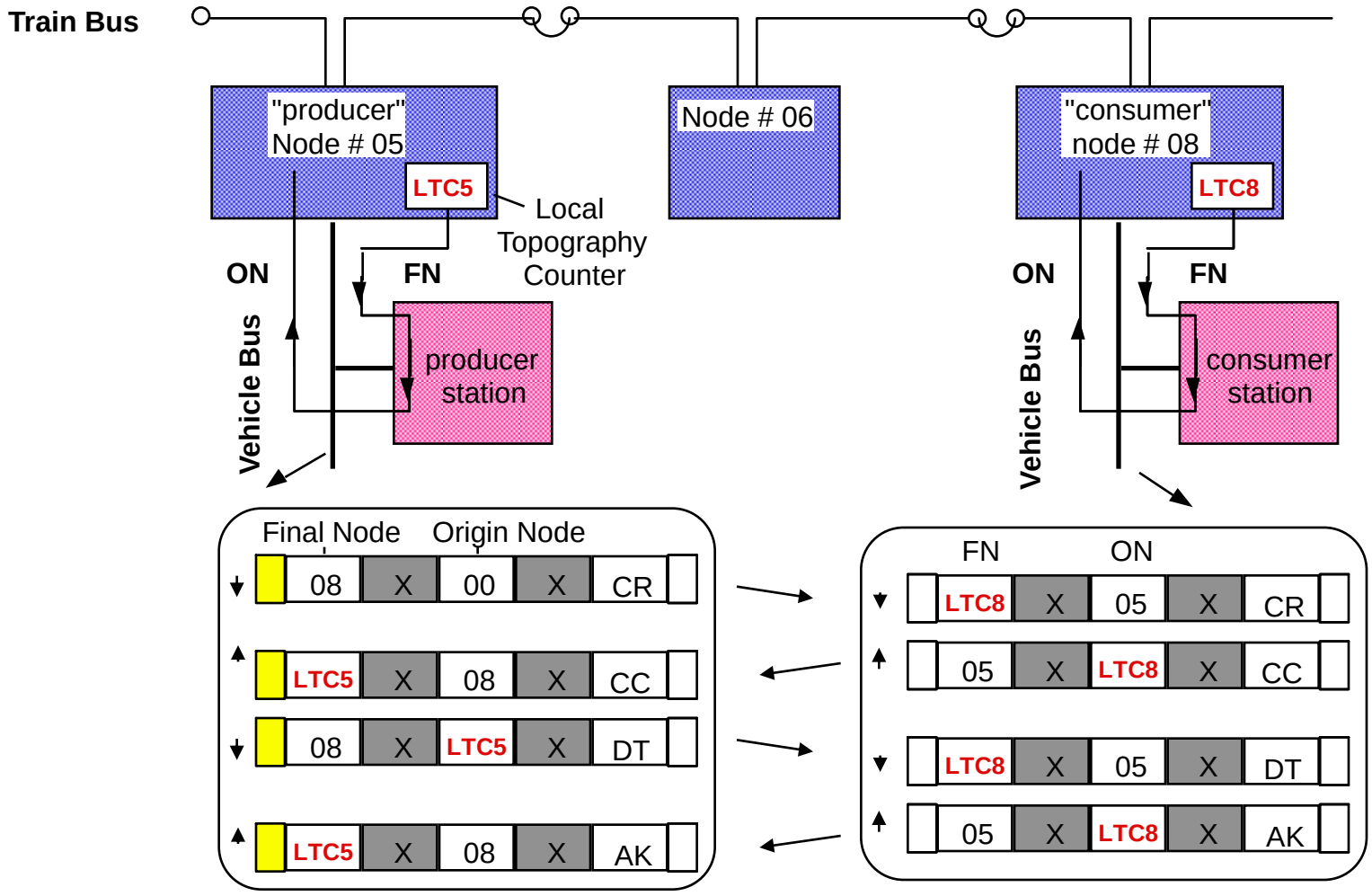
However, it is not possible to guarantee that the topography change did not affect the addresses of either partner.

Even if no node address changed, it is not possible to guarantee that the reason why this address was chosen still holds.

Even if the node address and function did not change, the application mapping may not anymore be correct, since it was based on some dynamic property of the node which changed in between (e.g. attended driver seat).

Therefore, communication must be cancelled indistinctly when the transport protocol becomes unable to guarantee correct delivery in all cases.

Network: Inclusion of Topography Counter



The gateway substitutes the topography counter in place of the node address

Transport and Session Protocol

1. General Principles
2. Variables
    1. Principle of cyclic Process Data broadcast
    2. Traffic Stores principle and implementation
    3. Process Variables and Datasets
    4. Software structure
    5. Application Layer Interface for Process Variables
    6. Networking
3. Messages
    1. Principle of Messages communication
    2. Link Layer Interface
    3. Networking and Routing
    4. Transport and Session Protocol
    5. Software structure
    6. Application Interface

Transport: Message Transport Protocol

The MTP opens a connection for each call message and closes it after the reply message has been received.

It is half-duplex (call and reply cannot take place at the same time)

It uses in each direction a sliding window protocol with a window size of 1…7 and positive acknowledgement (negative is also possible)

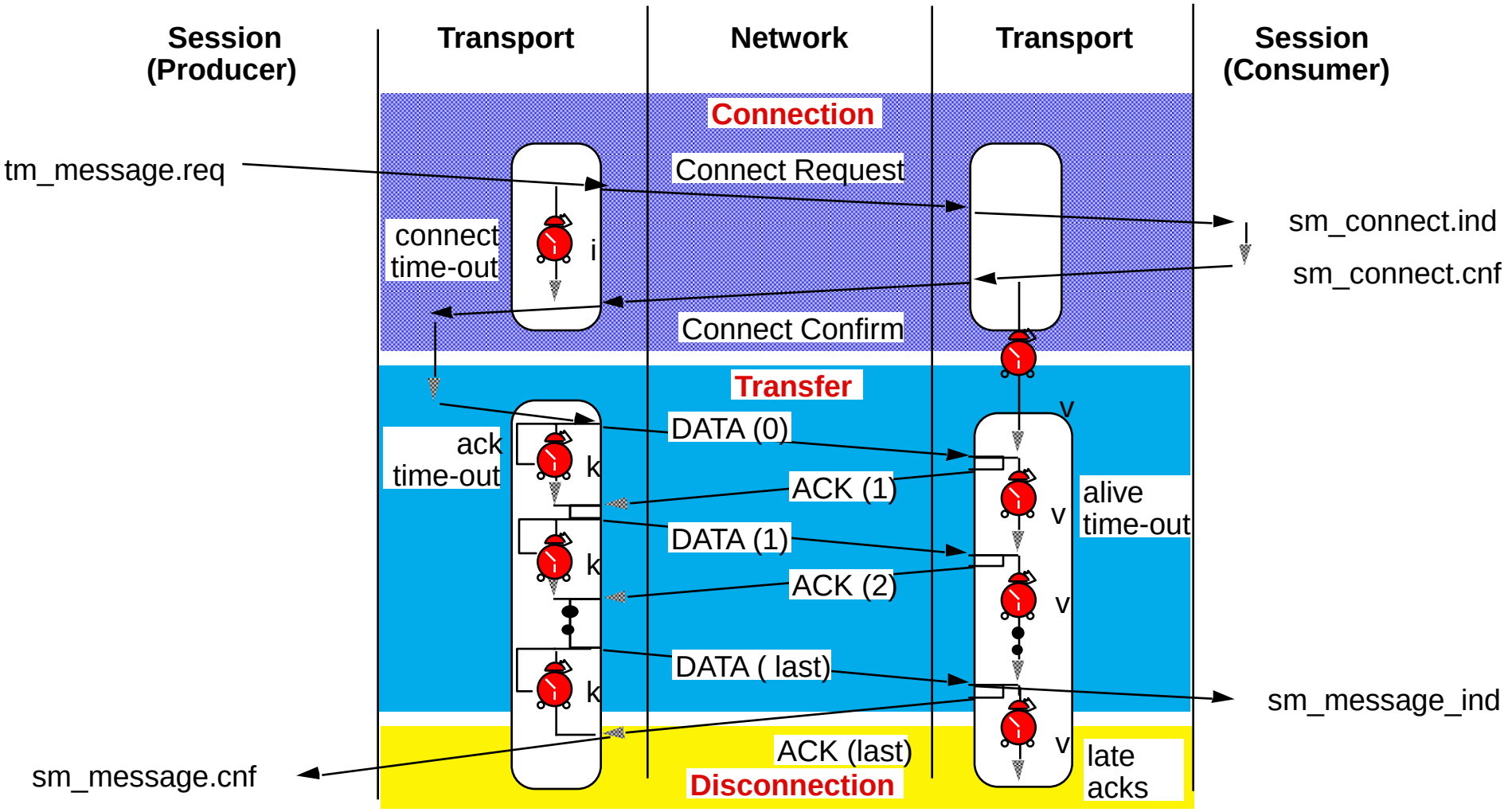
The frame and window size are negotiated at connection opening.

The origin and destination addresses uniquely identify the connection.

A connection reference prevents duplication of messages.

A caller reference pairs messages in a multi-tasking station.

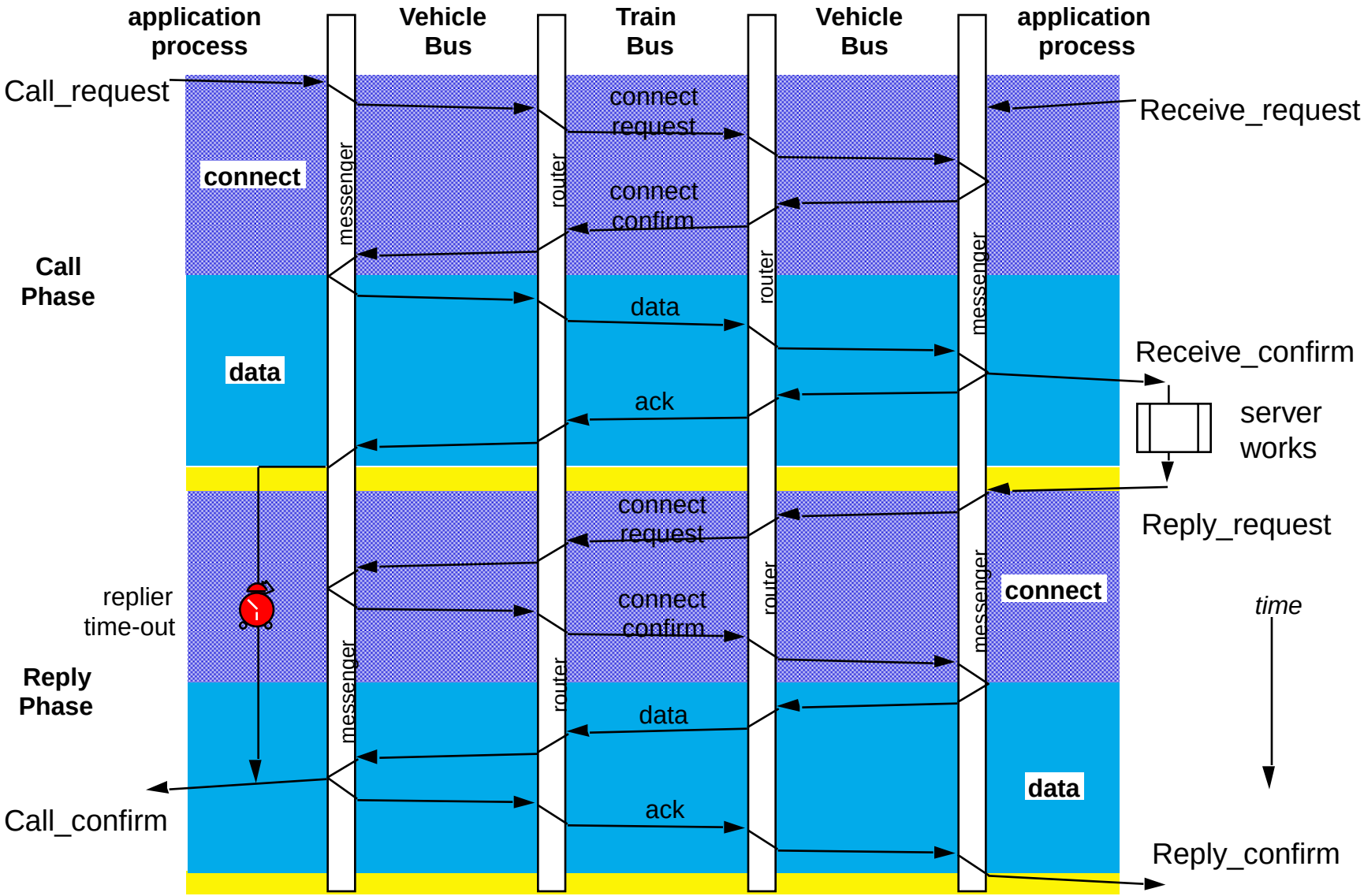
Transport: Frame Exchange in one Direction



A transport exchange consists of three phases: connection, transfer and disconnection

In this example, the transfer takes place with a window size of 1

Session: Call And Reply Over A Network



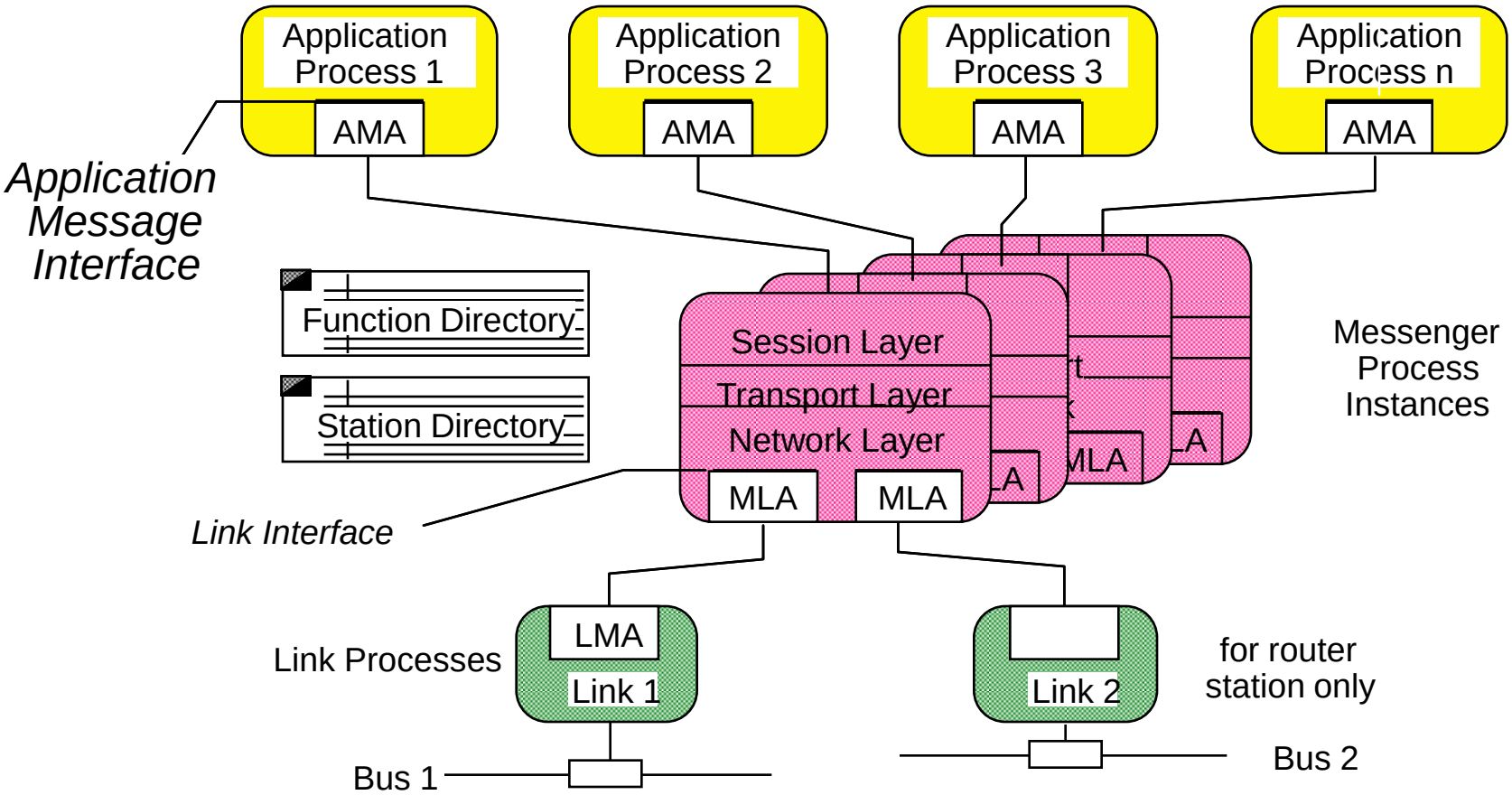
Software Structure

1. General Principles

2. Variables
    1. Principle of cyclic Process Data broadcast
    2. Traffic Stores principle and implementation
    3. Process Variables and Datasets
    4. Software structure
    5. Application Layer Interface for Process Variables
    6. Networking

3. Message Data
    1. Principle of Message Data communication
    2. Link Layer Interface
    3. Networking and Routing
    4. Transport and Session Layer
    5. Software structure
    6. Application Interface

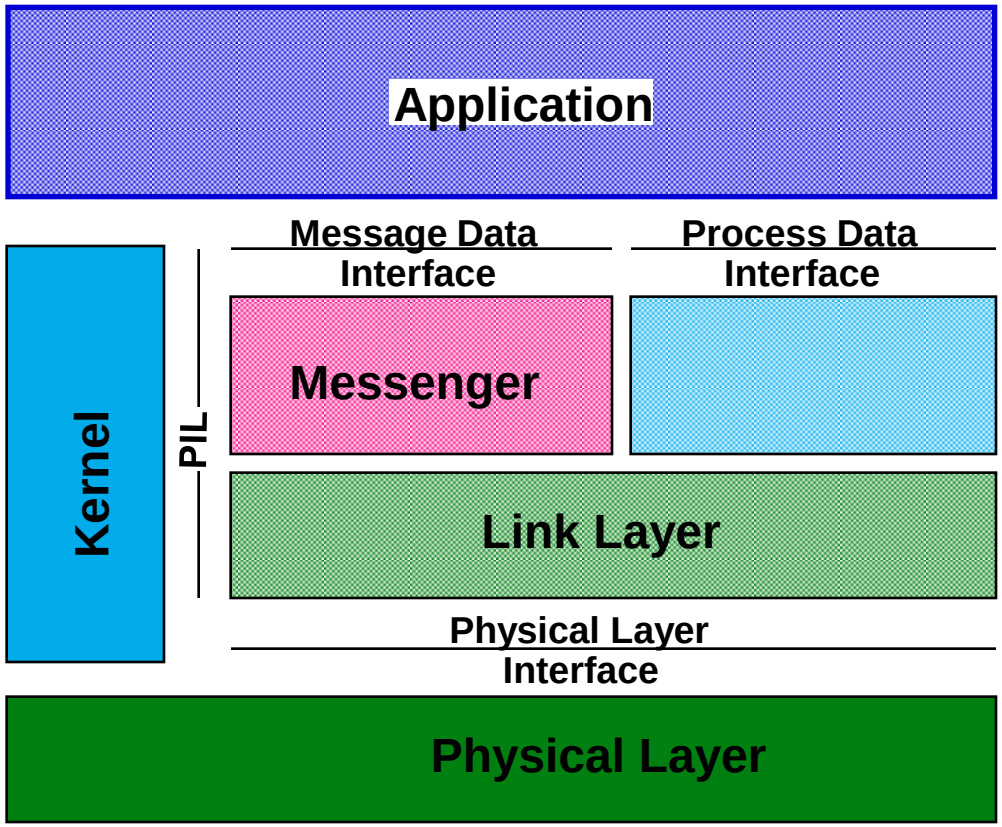
Software: Message Software Structure



Applications access the network through the Application-Message Interface (AMI)

This interface supports multiple simultaneous calls and replies

Software Interfaces



Porting of the TCN software to different platforms is eased by well-defined interfaces between the communication software and the application, the kernel and the link layer. The bus-specific link layer also has a defined interface to the physical layer.

Processor Interface Library (PIL)

The Messenger relies on services of the kernel
(e.g. memory allocation, timers, task wake-up)

To ease portability, the kernel services are defined in an interface module, the Processor Interface Library (PIL).

The PIL provides a set of basic functions as any commercial kernel (e.g. VRTX, WinWorks) can deliver.

Writing the PIL module is part of the porting process. In many cases, the PIL modules consists only of a redefinition of existing functions.

PIL Functions

Block copy	pi_copy8, pi_copy16	
Interrupt Control	pi_disable, pi_enable pi_call_hw_int	disables/enables all interrupts software trap
Dynamic Memory	pi_alloc, pi_free	allocate memory block, free it.
Queues:	pi_create_queue pi_accept_queue pi_send_queue pi_receive_queue	create queue, define priority check if a mesages is in the queu insert a message in the queue suspend until message or time-out
Semaphores	pi_create_semaphore pi_inquiry_semaphore pi_pend_semaphore pi_post_semaphore	create semaphore, define priority check semaphore value decrement sema, suspend if 0. increment semaphore
Tasks	*pi_create/delete_task* pi_lock_taks pi_unlock_taks	*not part of PIL* begin critical section end critical section
Time-outs	pi_create_time-out pi_enable_time-out pi_disable_time-out pi_delete_time-out	define function to call enable a time-out and specify value disable time-out delete time-out

AMI Procedures

Caller Interface

am_call_request	sends a call message
am_call_confirm	called on arrival of the reply message (or error)
am_call_cancel	cancel this call

Replier Interface

am_bind_replier	announces the service to the messenger
am_unbind_replier	retires the service
am_receive_request	expresses readyness to receive
am_receive_confirm	called on arrival of a call message
am_reply_request	sends the reply message
am_reply_confirm	terminates the reply
am_receive_cancel	cancels a receive or an unconfirmed reply

Directories (network layer access)

am_directory_insert	inserts an entry in the directory
am_directory_remove	removes an entry in the directory