A simplified classic CANopen to CANopen FD migration path using smart bridges

Christian Keydel, Embedded Systems Academy,
Olaf Pfeiffer, Embedded Systems Academy, Uwe Wilhelm, PEAK-System Technik,

Classical CANopen devices that are confronted with a CANopen FD message will produce an error frame. Therefore, classical CANopen devices and CANopen FD devices cannot directly share the same transmission media. This is a challenge when migrating existing classical CANopen systems towards CANopen FD.
The CANopen FD Smart Bridge introduced in this paper physically separates the classical CANopen and CANopen FD devices from each other. One port of the bridge runs in classical CANopen mode while the other runs in CANopen FD mode. The bridge physically separates the networks, but logically combines them to a single network.
Where possible, the smart bridge hides protocol specific details and allows a single CANopen (FD) manager to communicate and handle all devices connected, no matter if they are "classical" or "FD". Especially the service requests are converted from SDO to USDO and vice versa by the bridge.
This paper shows both, the possibilities but also the limits of smart bridging different CANopen protocol variants.

Challenges of CANopen and CANopen FD inter-operability

The new CANopen FD protocol has several advantages over classic CANopen: A much more flexible communication model that includes fully-meshed and broadcast communication with Universal Service Data Objects (USDOs), a higher potential bandwidth and larger messages that through the use of CAN FD also offer the extra space needed for authentication, for example with CANcrypt.

However, existing classic CANopen devices cannot be mixed with CANopen FD devices on the same network cable since classic CANopen devices will generate errors for any CANopen FD communication they detect.
So today, when designing new CANopen based networks or adding new functions, features, nodes or security to an existing CANopen system, you have the following options:

1. Stay entirely in classic CANopen
2. Do a complete transition to CANopen FD

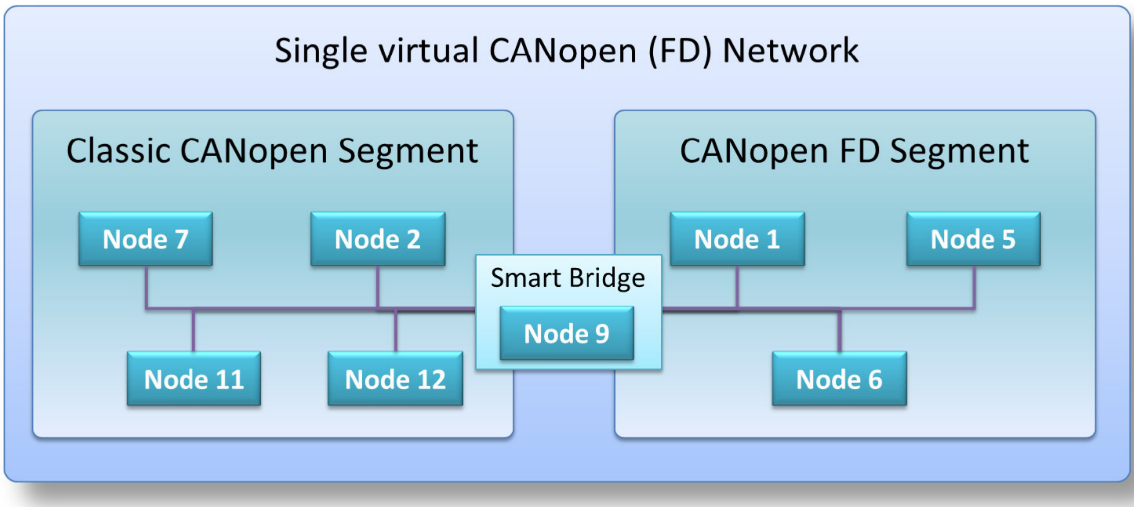

Figure 1: Connecting classic CANopen to CANopen FD using a smart bridge

3. Mix classic CANopen and CANopen FD using a smart bridge

For any development initiated today, you should not limit yourself to classic CANopen. Any new CANopen device you build today should be at least CANopen FD “ready” so that you can easily add enhanced features, faster updates and security “at any time”.

However, a complete transition requires that all devices connected are CAN FD capable. This typically requires new hardware designs or replacement of off-the-shelf CANopen with CANopen FD components and is only an option within the scope of a new system design or a complete redesign of an existing system.

To take best advantage of currently existing classic CANopen devices, you need an easy migration path from classic CANopen to CANopen FD.

Therefore, the third option is the best. It allows: a step-by-step transition from classic CANopen to CANopen FD or to add legacy devices and networks to a new CANopen FD design. The idea is that you use two network branches in your system: a classic CANopen branch with those devices without CANopen FD support and a CANopen FD branch with the newer devices that already support it. Making sure that all node IDs are unique, the two are connected as segments of a single virtual network, using a smart bridge as illustrated in Figure 1.

CANopen (FD) Smart Bridge requirements

Note that while there is no official standard for this type of bridge yet, we at Embedded Systems Academy saw the need for such a device to ease the transition to the emerging CANopen FD protocol, and, thus came up with its concept.

In order to keep system complexity minimal, the basic CANopen requirement of unique node IDs remains. In the complete system consisting of both CANopen and CANopen FD devices, each node ID must still be unique. If a node ID number 5 is used by a classic CANopen device, then no other CANopen or CANopen FD device in this system may use node ID 5.

Who is who and where?

In order to do anything “smart”, the bridge first needs to learn which nodes are connected.

- By monitoring the CANopen (FD) traffic like boot-ups, heartbeats and emergencies on both sides of the bridge, the bridge learns which nodes are connected to which port of the bridge. The devices are identified by their node ID.
- By monitoring the NMT Master message, the bridge learns where the NMT Master is located.

NOTE: the bridge should be up and running BEFORE any CANopen (FD) communication starts. In some systems it might make sense that on power-up the bridge transmits the NMT Master message with a reset request, ensuring that the bridge does not miss a boot-up message.

Default forwarding mechanisms

Once the bridge has the complete picture of the network segments and connected devices, CANopen (FD) messages can be “smartly” forwarded.

- Global unique messages such as NMT Master messages, bootup messages, SYNC messages and heartbeats are always forwarded to the other segment, as their content is the same in CANopen and CANopen FD and might be required by all devices connected.
- The EMCY (Emergency) messages are forwarded and converted into the respective format.
- The USDO and SDO requests and responses involving a node ID can be smartly forwarded to the destination segment.
- Per default, all PDOs with a length of eight bytes or less are forwarded.
- In an advanced mode, the bridge can
  - actively scan RPDO parameters to determine where consumers of a PDO are and
  - re-map PDOs if length is greater than 8 bytes

Smart USDO and SDO handling

The “smart” part of the bridge refers to the handling of the (Universal) Service Data Objects - (U)SDO. These are completely transparent for all nodes connected on either side.

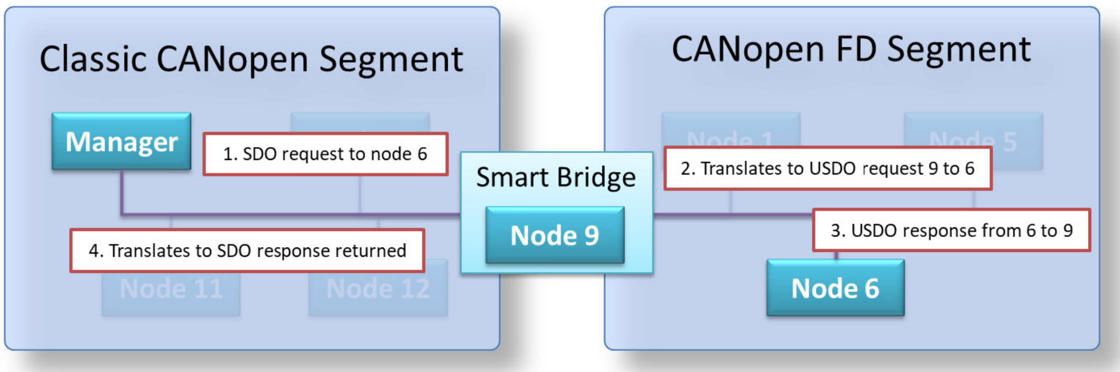

Figure 2: Handling SDO to USDO forwarding

If the default SDO client on the classic CANopen side sends an SDO request to a node on the CANopen FD side, the bridge translates it to a USDO request from the bridge itself to the node on the CANopen FD side. The USDO response is received and converted back to an SDO response on the classic CANopen side. These steps are shown in Figure 2.

If any CANopen FD device sends a USDO request to a node located on the classic CANopen side, the bridge translates this USDO request to a default SDO client request on the classic CANopen side. It waits for the SDO response and translates that back to an USDO response on the CANopen FD side as illustrated in Figure 3.

only this device may actively produce SDO requests. Note that there are mechanisms to support additional client channels organized by an SDO Manager, but due to the complexity involved, this is rarely used.

With the smart bridge, the preferred method is that the classic CANopen side has no device producing SDO client requests. This ensures that the bridge itself can use all default SDO client channels on this side. If another device is actively using the default SDO client channels, like a CANopen Manager present on the classic CANopen side, then the bridge prohibits that CANopen FD devices connected send USDO client requests to devices on the classic CANopen side.

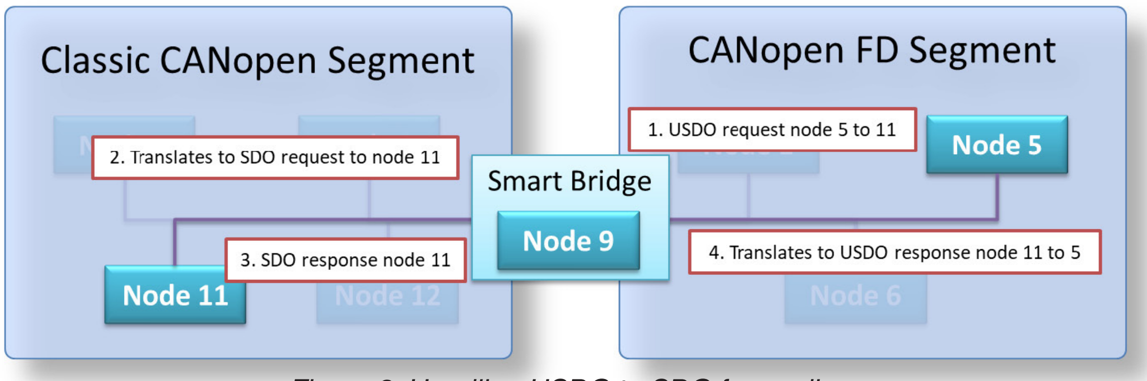
Classic CANopen Segment
2. Translates to SDO request to node 11
3. SDO response node 11
Node 11
Node 12
Smart Bridge
Node 9
CANopen FD Segment
1. USDO request node 5 to 11
Node 5
4. Translates to USDO response node 11 to 5
Node 6

Figure 3: Handling USDO to SDO forwarding

If you are familiar with CANopen (FD), you will see some challenges and limitations:

1. Potential default SDO client collision on classic CANopen side
2. Different segmentation size handling of SDO and USDO

In classic CANopen there is only “one set” of SDO client channels available. By default, these belong to the CANopen Manager, so

If SDO / USDO requests and responses only deal with transferred data up to four bytes, the bridging process is relatively simple. One pair of a USDO request and response is translated into a single SDO request and response and vice versa.

The handling becomes more challenging with segmentation. On the CANopen FD side, a USDO expedited transfer involving a single USDO request and response

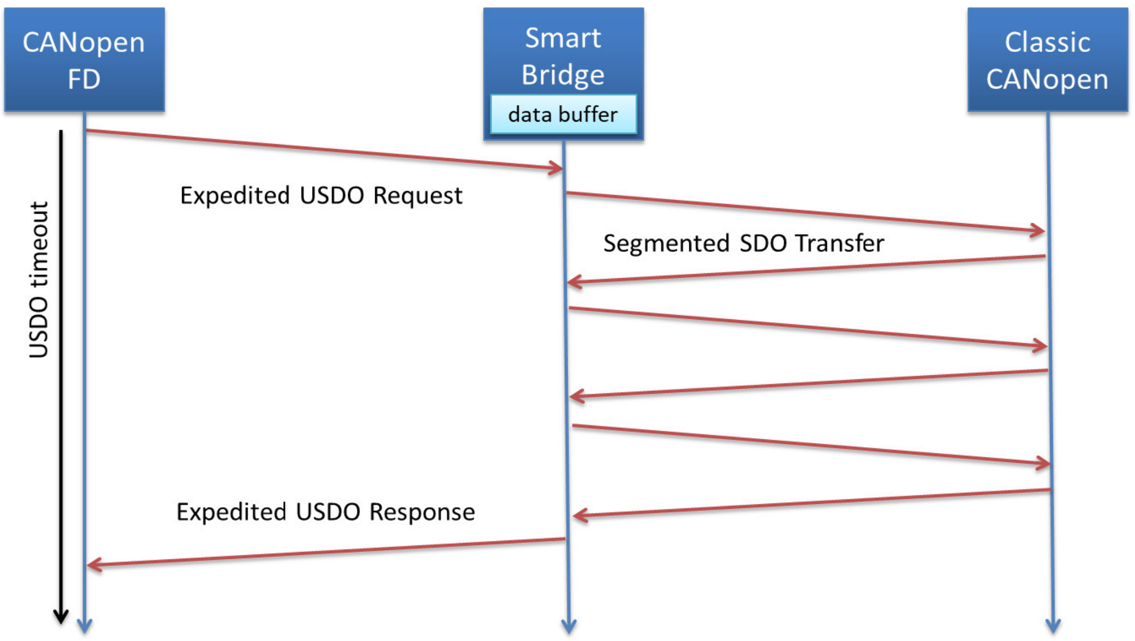

Figure 4: Expedited USDO transfer > 4 bytes translates to segmented SDO transfer

message can transfer up to 56 bytes of data. To convert such a transfer to classic CANopen, we need to use segmented or block SDO transfer. On the CANopen FD side, this also means that USDO requests send to devices in the CANopen segment will be slower, since they translate to multiple messages being exchanged on the classic CANopen side of the bridge. To cope with this situation, the bridge requires adequate buffers and the USDO timeout for the USDO client might need to be increased. Figure 4 illustrates this scenario for both read and write (upload and download) accesses.

Figures 5 and 6 show the opposite scenario: a classical CANopen Manager making segmented read and write accesses to a CANopen FD device. On read, the transfer on the CANopen FD side happens with one access, filling the buffer in the bridge. Afterwards the segmented transfer on the classical CANopen side continues. For a write transfer, the segmented portion of the transfer happens first on the classical side. Once the buffer in the bridge is filled, the single expedited transfer on the CANopen FD side starts.

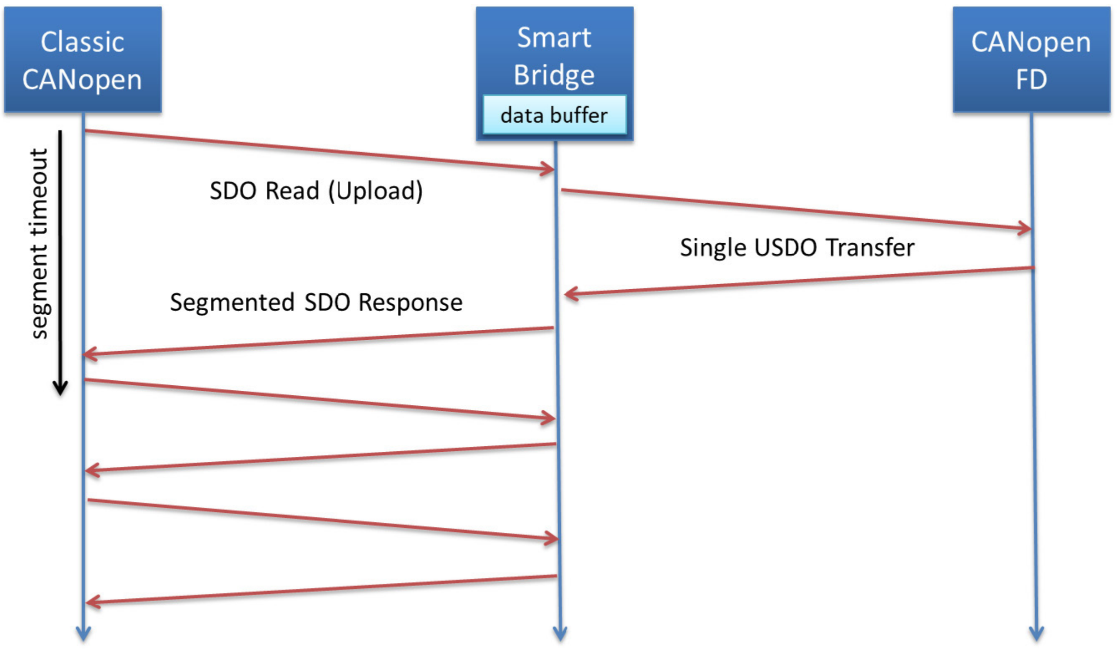

Figure 5: Segmented read (upload) from classical CANopen to CANopen FD

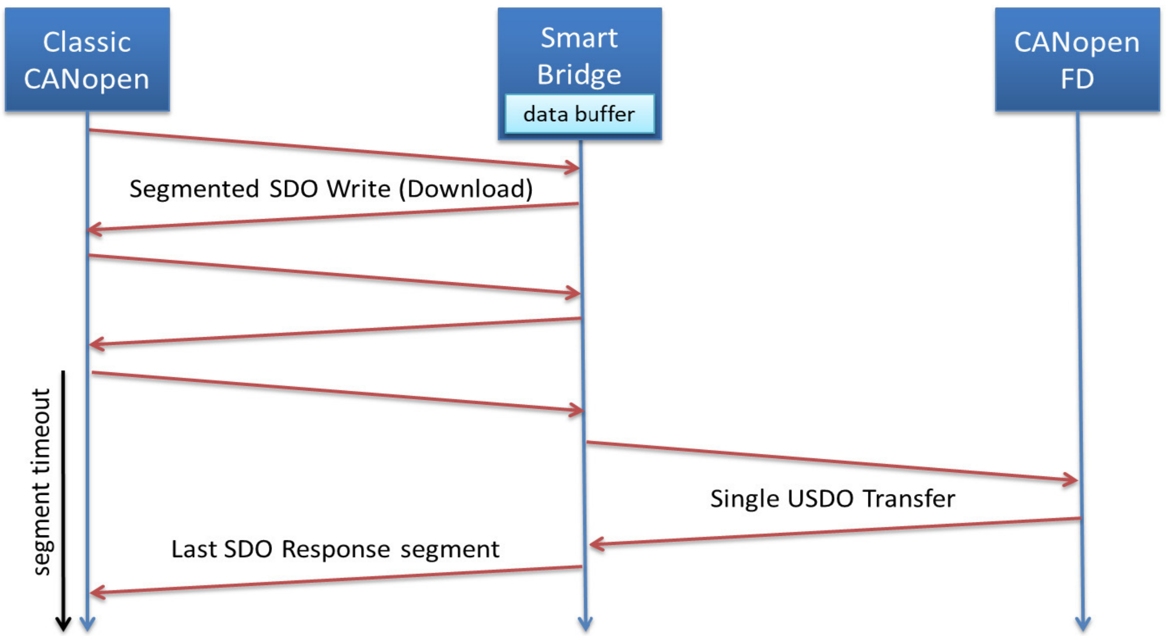

Figure 6: Segmented write (download) from classical CANopen to CANopen FD

In the case of an error resulting in an abort, the CANopen Manager will now see a slightly different behavior between devices in front of and behind the bridge: when communicating with a regular classical CANopen device in front, an abort like “Object does not exists or is not available” is generated by the server with the first segment transferred. For devices “behind” the bridge, such an abort will only be returned with the last segment of the segmented transfer. So, the downside here is, that the entire segmented transfer needs to be executed, before the server can confirm or abort the transfer.

Advanced PDO handling

For all cases where PDOs have a length of 8 or less bytes, the bridge can do a smart forwarding, if it knows which PDOs are consumed by which devices. In CANopen (FD) this information is provided by all devices in the RPDO Communication parameters. If enabled to do so, the bridge can actively read these parameters to determine which RPDO COBID (CAN identifier) are consumed by which device.
Once an appropriate forwarding table for PDOs has been determined, it should be saved in non-volatile memory, as this scanning can be a lengthy process (worst case some 512 SDO requests per node found).

When it comes to handling PDOs with more than 8 bytes, then there are currently no automated processes available, as multiple challenges arise:

1. How to map a single PDO with more than 8 bytes on the CANopen FD side to multiple PDOs on the classical CANopen side?
2. How to handle triggering if multiple PDOs on the classical CANopen side need to be converted to a single CANopen FD PDO?

As these settings are highly system and application specific, they must be made through manual configuration.

Advanced SYNC handling

When it comes to the SYNC signal, the default setting is, that the bridge also forwards this signal from whichever side it is produced to the other side. However, each bridge adds a slight delay. A message first needs to be fully received, before it can be forwarded.
For highly synchronized systems it will shorten delays, if the SYNC signal is produced directly by the bridge, synchronously on all ports. There can still be delays on individual ports (for example if CAN controller is busy receiving a frame), however, statistically the delays will be smaller in comparison to a pure forwarding based method.

Bridge Implementation and Performance

The first Smart Bridge implementation has been made on a platform provided by PEAK-System Technik GmbH. Their PCAN-Router FD features an NXP LPC4000 microcontroller based on ARM Cortex-M4 running at 120 MHz with two CAN (FD) ports.
Measurements have shown that pure bridge processing times for individual messages are typically below a few single microseconds. Therefore, messages that can be directly forwarded only have a minimal delay: typically the message transmission time (entire message needs to be received, before it can be forwarded) is bigger than the delay by the bridge.
As soon as segmentation translation is involved, delays depend on the response times of the nodes on the other side of the bridge. As an example, the biggest USDO expedited transfer can have up to 56 bytes of data. The classical segmented transfer requires 8 segments for that amount of data. Depending on the performance of the device, individual segments may use a few single milliseconds to multiple tens of milliseconds. As a result, the entire segmented transfer can use in the range of below 50ms all the way to hundreds of milliseconds.

Outlook

In this paper, we limited our view to a single bridge with two ports. However, it should be noted that
1. you could use multiple bridges in one system resulting in 3 or more segments or
2. one could envision bridges with more than 2 ports
This would allow even more flexible system configurations as shown in Figure 7. As each bridge also allows to extend a network physically, a smart combination of such bridges with both classical CANopen and CANopen FD allows the integration of larger systems with more efficient use of the CAN (FD) communication channels.

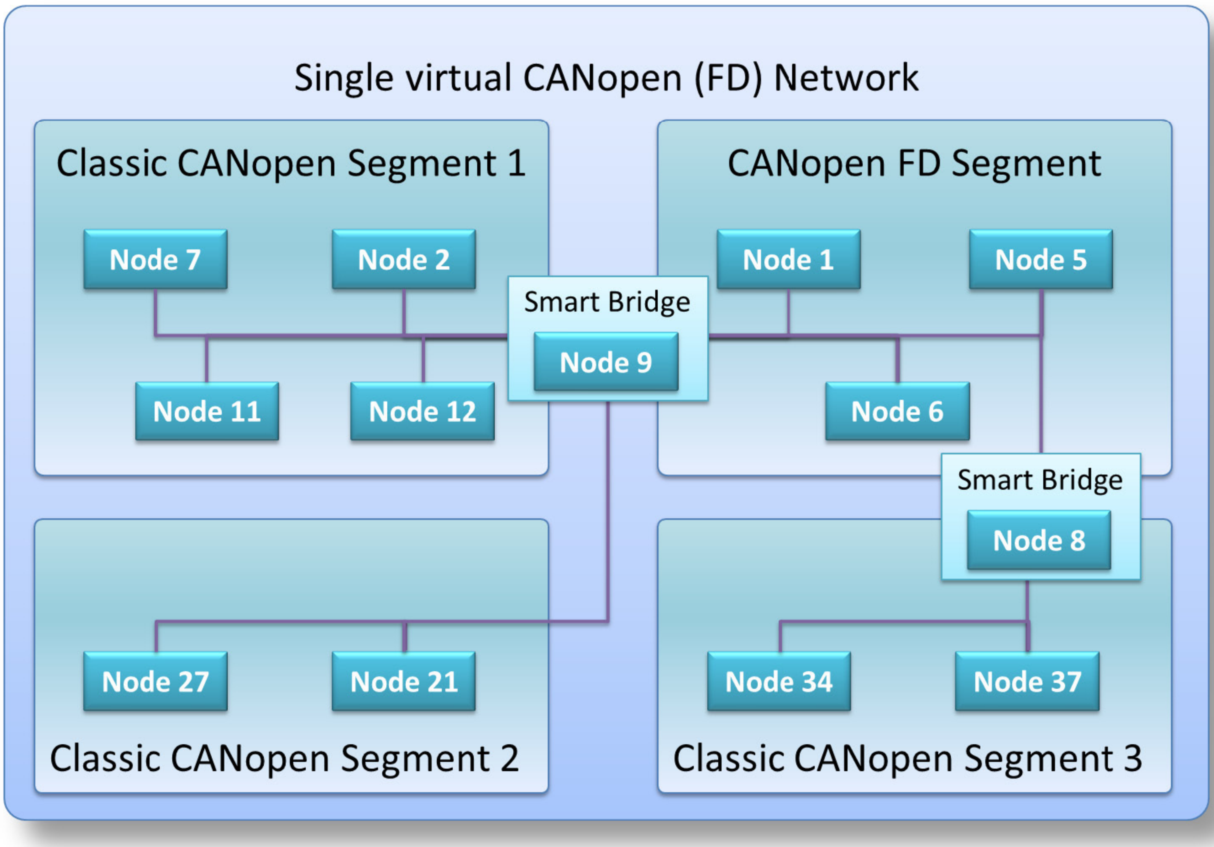

Figure 7: Segmented read (upload) from classical CANopen to CANopen FD

Christian Keydel
Embedded Systems Academy GmbH
Bahnhofstr. 17
DE-30890 Barsinghausen
www.em-sa.com

Olaf Pfeiffer
Embedded Systems Academy GmbH
Bahnhofstr. 17
DE-30890 Barsinghausen
www.em-sa.com

Uwe Wilhelm
PEAK-System Technik GmbH
Otto-Röhm-Str. 69
DE-64293 Darmstadt
www.peak-system.com