Relation of throughput and latency in CAN and PLCA networks

Christoffer Mathiesen, (Kvaser)

The throughput of a simple point-to-point data link is dependent on two things: the link-speed and the relation between payload and overhead. A multidrop network adds the complication of having to factor in waiting for access to the shared medium. Classical CAN, CAN-FD and CAN-XL comes with built-in multidrop support and a communication protocol beneficial to this kind of structure. 10BASE-T1S and 10BASE-T1M using PLCA or D-PLCA aims to solve the access problem, but for a multidrop Ethernet bus. As these protocols all try to solve a similar problem, comparisons between them can assist a system designer decide which to use. But as they are implemented similarly yet differently it can be difficult to discern which protocol fit better than another. This paper aims to describe a couple of scenarios and show why one protocol may be better suited than another in those cases.

PLCA and D-PLCA behavior in short

10BASE-T1S can utilize the sub-protocol PLCA to achieve bus behavior with real-time characteristics. The yet unreleased version 10BASE-T1M, which is a further development of 10BASE-T1S, can imple-ment both PLCA and D-PLCA. With these sub-protocols nodes get access to a trans-mission slot in a round-robin fashion [1]. These slots are not of fixed length in time, and if a node does not have anything to transmit the next node in the order can quickly get an opportunity instead. If a node does have data to transmit the next node in the order will have to wait until the sending node has finished.

A PLCA network needs to be configured at design-time. One node is configured as the beacon, which all other nodes will sync to. Since the network is known, nodes will not have to back off from transmitting data. The throughput will vary only by how much the node will have to wait for other nodes to do their transmissions. The PLCA protocol allows a network to have 2-255 nodes, each of which has an opportunity to send data during each loop of the round-robin cycle [1]. In practice, since 10BASE-T1S is not designed for more than 8 nodes, having more nodes than such may be difficult to achieve. 10BASE-T1M is being designed to handle 16 nodes.

D-PLCA aims to alleviate the need for the network to be known at design-time by making it possible to dynamically add and remove nodes to the network. Because of this dynamic nature, the need for the designer to assign a node as the beacon is not necessary as the nodes themselves can mantle the role if required. To accommodate new nodes to the network, there must always exist an unused slot. The default for the protocol is to always have 8 slots minimum, whether used or not, and potentially grow from there [2]. Because of the round-robin style order and the possibility of adding nodes to an active network, a node may collide with another node during a transmission slot. This requires the colliding nodes to back off from transmission and reassign themselves to another slot in the round-robin order and try again later [2].

Scenario setup

To be able to make comparisons of the protocols the setup of each needs to be described. An overview can be seen in Table 1. For protocols that use variable bitrate, two speeds are listed. A CAN network can potentially be of any size, whereas PLCA and D-PLCA are limited to a maximum of 255 (in practice up to 8 for 10BASE-T1S and 16 for 10BASE-T1M). Because the number of nodes in the network impacts

the throughput of PLCA and D-PLCA, and that the minimum slots in D-PLCA is 8, and the last one of these slots is not to be in use [2], the amount of network nodes for each protocol is set to 7. For both PLCA and D-PLCA it is assumed that burst mode (multiple ethernet frames within the same transmission slot) is disabled.

The CAN family of protocols use dynamic stuff-bits to ensure transmission integrity, but as these bits are dependent on the pattern of the transmission, an assumption of how often this happens must be made. For this reason, a stuffing-factor of 8% is chosen, as this corresponds to the likelihood of a stuff bit given a completely random sequence of bits. CANXL always put a stuff bit every 10 bits in the data phase, and this is represented by a second value in the stuff factor column. 10BASE-T1S/T1M does not use stuff bits, so PLCA and D-PLCA stuffing factor is 0 %.

Each protocol may send different payload sizes per packet. Because BASE10-T1S/T1M is used to send Ethernet frames, the minimum payload size for both PLCA and D-PLCA is 46 bytes [1]. CANXL has a minimum of 1 byte payload. For Classical CAN (CANC) and CANFD, no extended frames are used. For each of the protocols, it is assumed that the other nodes in the network are using the same protocol.

While bus design is not discussed in this paper, it is important to note that the CAN protocols are able to use many different link-speeds other than those used in the scenarios which are to be described. CANXL can potentially even achieve speeds of 30 Mbit/s! This gives leniency in the requirements of the bus's design if high link-speeds are not needed, but better capabilities if they are. 10BASE-T1S/T1M can only run at 10 Mbit/s, and the bus must always be designed with this bitrate in mind. Because it is possible to run at a faster rate as well as the same as 10BASE-T1S/T1M, CANXL gets two entries. 10 Mbit/s to be able to have comparable numbers to PLCA and D-PLCA, and 20 Mbit/s to show the difference a faster link-speed makes.

Table 1: Settings used for the different protocols.

Protocol	Speed (kbit/s)	Nodes	Stuff factor	Payload (byte)
CANC	1000	7	8 %	0-8
CANFD	1000/8000	7	8 %	0-64
CANXL	1000/10000	7	8 % / 10%	1-2048
CANXL (20 Mbit)	1000/20000	7	8 % / 10 %	1-2048
PLCA	10000	7	0 %	46-1500
D-PLCA	10000	7	0 %	46-1500

Scenario 1: Single bus user, no data

The simplest scenario is that a single node continuously sends messages containing minimum data while no other node ever tries to send packets. Because this test only considers the packets and not the data within them throughput must be calculated on the messages themselves. In this scenario, the CAN protocols can repeatedly get access to the bus and send the messages, leading to effectively 100% bandwidth utilization. Despite faster link-speed, CANXL's latency is worse than CANFD's mainly because of the increased overhead caused by new data fields in the header and footer.

PLCA and D-PLCA suffers significant overhead from the protocol when only one node use the bus and the payloads are of these small sizes. Because of the protocol overhead in this scenario PLCA and D-PLCA can only utilize slightly more than 70% of the available bandwidth.

Table 2: Average values when only a single node continuously sends the smallest possible packets.

Protocol	Payload (byte)	Latency (µs)	Bus util. (%)	Eq. Thru-put (kbit/s)
CANC	0	51.0	100%	1000.0
CANFD	0	35.1	100%	1822.0
CANXL	1	49.5	100%	3454.6
CANXL (20 Mbit)	1	42.8	100%	4000.0
PLCA	46	85.1	73.9%	7391.3
D-PLCA	46	88.4	71.2%	7115.4

Scenario 2: Single bus user, full data

Another simple scenario is that a single bus node repeatedly sends a message containing as much data as possible. In these cases, the older CANC and CANFD protocols suffer a lot in throughput from their relatively large overhead in each packet.

CANXL fares better and is on a comparable level to both PLCA and D-PLCA. Despite equal bit-speed to PLCA and D-PLCA, CANXL falls behind in absolute throughput due to the stuffing procedure and slow link-speed in the CAN compatible phases. Utilizing a higher link-speed with CANXL compensates for these factors and enables throughput not possible using 10BASE-T1S/T1M.

Table 3: Average values when only a single node continuously sends the largest possible packets.

Protocol	Payload (byte)	Latency (µs)	Time Data (%)	Eq. Data Thruput (kbit/s)
CANC	8	120	53.3%	533.3
CANFD	64	104.3	61.4%	4911.3
CANXL	2048	1850.8	88.5%	8852.4
CANXL (20 Mbit)	2048	940.4	86.8%	17367.0
PLCA	1500	1248.3	96.1%	9613.1
D-PLCA	1500	1251.6	95.8%	9587.7

Scenario 3: Single bus user, common data payloads

To compare the protocols directly against each other they must be used in the same manner. If a payload of 64 bytes is to be sent all protocols except for CANC are able to send it in a single message. The multiple messages combined with a comparably slow link-speed, CANC is almost a magnitude slower than the rest. The large number of packets needed also introduces greater risk of increased and variable latency, as other nodes may interlace their transmissions in between the large payload.

Table 4: Results when a single node is to deliver a payload of 64 byte without interference from other nodes

Protocol	Payload (byte)	Tot. Msgs	Tot. time (µs)	Eq. Data Thru-put (kbit/s)
CANC	64	8	960.0	533.4
CANFD	64	1	104.3	4911.3
CANXL	64	1	104.9	4880.8
CANXL (20 Mbit)	64	1	70.5	7267.6
PLCA	64	1	99.5	5145.7
D-PLCA	64	1	102.8	4980.5

If a large single payload needs to be transmitted multiple packages are needed to complete the transmission. In this example, a file of 4kB is to be sent. (Table 5) CANFD's small maximum payload of 64 byte leads to significant overhead, which is the reason why it performs worse. The newer CANXL is almost on par with PLCA and D-PLCA but has noticeable lower throughput. Despite having equal link-speed in the data phase and one fewer message sent, CANXL's implementation of stuffing in the data phase and slow speed CAN-compatible negotiation phase are the culprits behind the almost 800 kbit/s lower equivalent data throughput when the link-speed is the same as for PLCA and D-PLCA. As seen in Scenario 2, increasing the link-speed can compensate for the induced overhead.

Table 5: Results when a single node is to deliver a payload of 4096 byte without interference from other nodes.

Protocol	Payload (byte)	Tot. Msgs	Tot. time (µs)	Eq. Data Thruput (kbit/s)
CANC	4096	512	61379	533.4
CANFD	4096	64	6695.5	4911.3
CANXL	4096	2	3700.7	8854.5
CANXL (20 Mbit)	4096	2	17375.7	17375.7
PLCA	4096	3	3399.5	9639.1
D-PLCA	4096	1251.6	3406.1	9620.4

Figure 1 and 2 shows the same test done for a couple of different payload sizes. For small payloads, CANFD and CANXL have higher throughput, but after 64 byte they lose ground PLCA and D-PLCA. CANFD plateau at slightly below 5 Mbit/s after 64 bytes, due to the maximum payload of 64 bytes of data per packet. CANXL sticks close to PLCA and D-PLCA but is noticeably behind unless a higher link-speed is used.

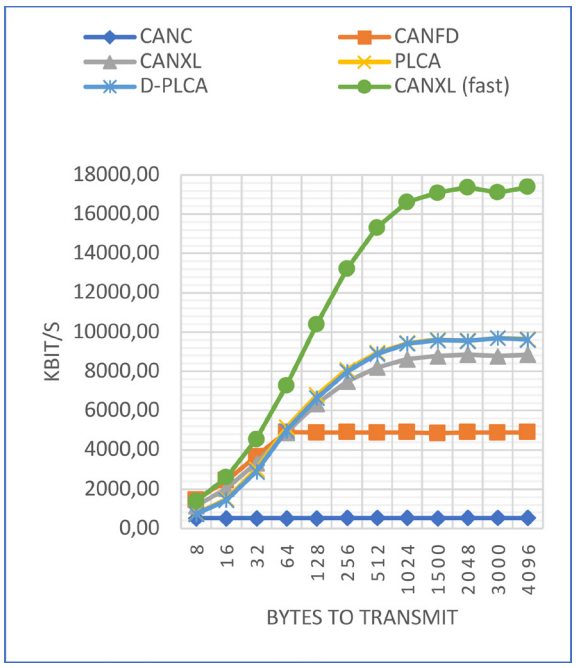

Figure 1: Throughput values for different amounts of bytes to transmit.

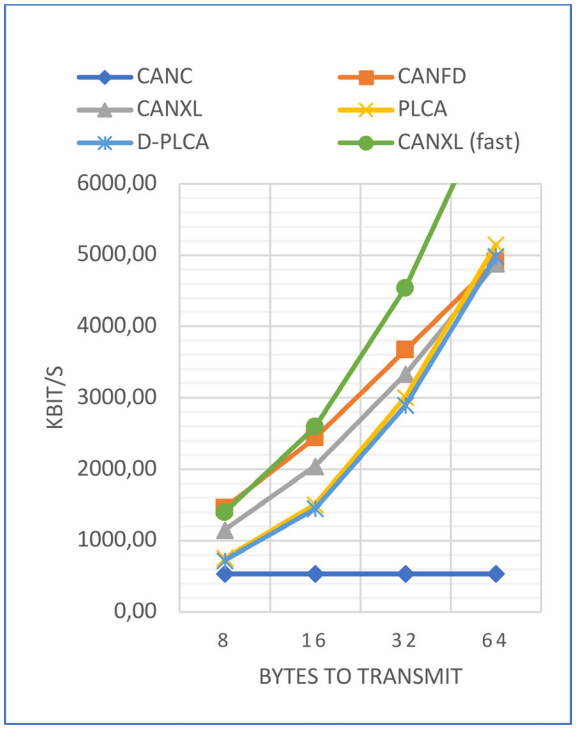

Figure 2: Throughput values for different amounts of bytes to transmit, zoomed in to first four values.

Scenario 4: Best- and worst-case latency of a message

A bus system is usually used with a real-time system, and as such the latency of packets can be of great importance. The CAN-protocols', PLCA's and D-PLCA's different implementations impact the latencies of packets and thus the responsiveness of a system.

As CAN can prioritize the bus-access via the arbitration phase, in the best case, a node can be guaranteed to get access to the bus as soon as it is free. However, although very unlikely, this has the potential to block out access to the bus entirely if a node is never able to win a slot in the arbitration phase.

In table 6 the different CAN protocols' different latencies are listed in a scenario where each node is to send a 64-byte packet. "Best Data" is when the bus was not in use, and transmission can start immediately. "Just Missed" means that the node just missed to participate in arbitration, and thus must wait until the other node currently using the bus has finished its transmission. "Wait All Send" means that the node has to wait for all other 6 nodes to send a 64-byte message. Usually this scenario is unlikely to occur in a well-designed CAN-system, but in order to compare to how PLCA and D-PLCA behaves this case is included.

PLCA and D-PLCA do not have any ability to prioritize messages, and all nodes must wait for its opportunity-slot for transmission. If the number of nodes is large, and all other nodes sends much data, this wait can potentially be very long. This wait cannot be infinite since each node only has so much time to send data during its opportunity-slot. Because D-PLCA is dynamic, this protocol could potentially end up in a situation where a node gets blocked from transmitting during its slot and must wait for an additional loop of the round-robin order, where it now has the last slot in the order.

In table 6 PLCA's and D-PLCA's different latencies are listed in a scenario where each node is to send a 64-byte packet. "Best Data" and "Best No Data" is when the message has been committed to be sent just before the node got its transmission opportunity, thus not needing to wait for access to the bus. "Just Missed" is when a node just yielded its transmission opportunity and must wait for the next opportunity. To be comparable to the CAN protocols, during the wait, one

other node also sends a 64-byte packet. “Wait All Send” is when a node just yielded its transmission opportunity and must wait for the next (all 6 other nodes use their opportunities to send a 64-byte message this transmission cycle).

Table 6: Different latencies for transmission of a packet given different scenarios.

Protocol	Best No Data (µs)	Best Data (µs)	Just Missed (µs)	Wait All Send(µs)
*CANC	64	8	960.0	533.4
CANFD	64	1	104.3	4911.3
CANXL	64	1	104.9	4880.8
CANXL (20 Mbit)	64	1	70.5	7267.6
**PLCA	64	1	99.5	5145.7
**D-PLCA	64	1	102.8	4980.5

* *CANC use 8-byte data packets. Other protocols use 64-byte packets.*
** *No Data still has a 46-byte data payload*

Discussion of the scenarios and protocols

From the scenarios previously outlined one can see that the protocols sometimes perform similar, and sometimes very differently. From the perspective of just sending small packets of a few bytes, despite the lower bitrate, the CANFD protocol beats all other protocols. Beating PLCA and D-PLCA is not too strange in these cases, given that the smallest Ethernet frame contains 46 bytes of data and unused bytes in the data field essentially becomes overhead. When the packet sizes increase to match this minimum, CANFD’s advantage quickly dissipates.

For transmission of large quantities of data, a link with higher throughput is always preferable. CANC and CANFD performs poorly in these cases because of their small payload sizes and comparably low link-speed. While CANXL, PLCA and D-PLCA all share the same link-speed of 10 Mbit/s, because of CANXL’s implementation of forced stuffing every 10 bits during the data phase, the actual throughput during this phase is only 9/10ths of the link-speed. Combined with the slow CAN-compatible phases of the protocol, and the usable data throughput of a CANXL link cannot in any scenario exceed ~8.9 Mbit/s, compared to PLCA and D-PLCA having a throughput of ~9.6 Mbit/s. It is important to note that if the bus is well designed CANXL can run at link-speeds higher than 10 Mbit/s. If a higher link-speed is used CANXL can beat both PLCA and D-PLCA in throughput.

When a bus is under heavy load the message latency potentially goes up. To be able to make comparisons between the protocols, in Scenario 4’s “Wait All Send” values it was assumed that the CAN protocols would be delayed equally as often by other nodes as in PLCA and D-PLCA. This comparison puts the CAN-protocols at a disadvantage since it ignores both the main benefit and drawback of CAN. A node can send both subsequently or be starved by the other nodes given if it wins or lose the arbitration in a standard CAN network. Because of this it cannot have a known latency other than the transmission latency. For important/critical messages, however, a well-designed CAN system could implement packets that are guaranteed to be sent as soon as possible. But in order to have comparable values in the table the choice of ignoring these traits was made.

Since PLCA runs on a round-robin system, and because burst-mode is disabled in the above-described scenarios, one node cannot send two packets without waiting for the system to loop back around to allow for transmission of the second packet. This greatly increases the minimum latency. But at the same time, because the system implements round-robin, the latency of any given packet can be guaranteed within a lower and upper bound. Unlike CAN, there is no way to implement a certain packet that’s guaranteed to be sent as soon as possible nor is it possible to starve out a node from ever accessing the bus.

The differences in real-time characteristics between the protocols has implications on the real-time design choices of a system implementing either type of protocol, but details of such implications is beyond the scope of this paper.

Conclusion

When it comes to latency, CAN buses are usually more responsive than a corresponding 10BASE-T1S/T1M system, but not in all cases. If packet sizes are small and with no to few collisions, CAN has very low latency. If packet sizes increase this advantage vanishes.

For any bus, if it is under heavy load the latency of any one packet is likely to increase. Since PLCA and D-PLCA lack any possibility to prioritize packets, if one is designing a system that needs to have prioritized packets sent as soon as possible, the CAN protocols are more beneficial as the potential latency for PLCA and D-PLCA can be very long. To be able to benefit from shorter response times for important packets care and thought must be made during the system design process.

Given a bus with a link-speed of at most 10 Mbit/s PLCA and D-PLCA systems running on 10BASE-T1S/T1M will always have a higher system-wide throughput than what any CAN system is capable of, though CANXL comes close in performance. The main cause for CANXLs performance falling behind PLCA and D-PLCA is because of the need to add stuff-bits. If the bus is well designed and capable of link-speeds of 20Mbit/s and beyond, CANXL can compensate for this overhead and achieve throughput not possible with 10BASE-T1S/T1M.

If one is looking for a multi-drop bus protocol for a system that will handle transmissions of large payloads, both PLCA and D-PLCA are good alternatives, with CANXL as a close second given that the bus structure is unable to achieve link-speeds greater than 10 Mbit/s. If higher link-speeds are possible CANXL can be the best in throughput as well. A system that will handle small packets would benefit mostly from CANFD, with CANXL as a close second.

Because CANXL is backward compatible with both CANC and CANFD, a node that can send CANXL packets is most likely capable of sending CANC and CANFD packets as well. To maximize both responsiveness and throughput, a well-designed system would utilize CANFD packets when transmitting small packets, and CANXL packets when transmitting large packets.

References

[1] IEEE Std 802.3™-2022
[2] Dynamic PLCA Node ID Assignment, https://www.ieee802.org/3/da/public/110420/beruto_3da_01_110420.pdf

Christoffer Mathiesen
Kvaser
Aminogatan 25A
SE-43153 Mölndal
christoffer.mathiesen@kvaser.com
www.kvaser.com