Congestion control in building automation networks: Considerations for KNX

Georg Neugschwandtner and Wolfgang Kastner
Automation Systems Group, Inst. of Computer Aided Automation, Vienna University of Technology
Treitlstraße 1-3, 1040 Vienna, Austria
{gn,k}@auto.tuwien.ac.at

***Abstract*—Current standard network protocols for distributed building automation were designed before the advent of the cost efficient fast networking technologies that are popular today. While these technologies are of great benefit for the network backbone, the possibility of "fast" nodes on such media overwhelming "slow" nodes or network segments increasingly comes into focus. Besides discussing the issue on a generic level, this paper closely examines the situation for the KNX protocol standard and suggests possible ways of improvement.**

I. Introduction

Building automation is concerned with automatic control and central supervision of building services equipment (in particular, equipment related to heating, ventilation and air conditioning and lighting). It is characteristic to this field of application that while throughput requirements on the automation network are low at the field level (sensors, actuators and low-level human-machine interfaces and controllers), central supervisory stations may need to handle considerable data rates due to the size of the entire system.

Traditionally, distributed building automation systems were implemented using a hierarchy of specialized, separate networks (field, automation and management level) to address this issue. With processing and communication getting cheaper over time, the possibility appeared to use a single network protocol throughout the system. All open protocol standards relevant for this purpose (BACnet [1], [2], CNP/LonMark [3], [5] and KNX [6], [8]) support network segmentation and routing, a necessity to cope with the potentially high device count and large distances covered in an integrated system.

Still, requirements remain diverse, with demands on throughput increasing (and opposing ones such as lowest cost decreasing) towards the root of a usually tree shaped network topology. This results in the ongoing trend of combining a high speed network backbone with low speed fieldbus segments. Ideally, protocols support various communication media to suit these purposes. The above mentioned standards provide this capability, with the choice of supported media reflecting their intended deployment focus.

Some time after these standards were written, IP networking (in particular over Ethernet) became pervasive and cheap. With commercial grade components sufficiently supporting the dependability and performance requirements of building automation networks, this development was bound to have considerable impact. To tap this source of cheap high bandwidth backbone infrastructure (and of course, to enable remote access via the Internet), tunnelling mechanisms were provided (e.g., [4] and [9]) and quickly got popular.

Leaving the original network layer in place ensured backward compatibility. However, this layer was designed without the unprecedented performance disparity between network segments in mind that exists today. Removing the backbone bottleneck may thus have undesired side effects. Network traffic does not only travel towards the root of the network hierarchy. Devices placed on a high speed backbone can quickly generate a high number of messages destined for lower levels. For example, a SCADA server could want to initialize its process image, resulting in status requests being sent to each sensor and actuator. Likewise, a supervisory controller may activate a complex scenario involving a large number of actuators, or an engineering tool could attempt downloading configuration data to multiple devices in parallel. Without appropriate countermeasures, all this very soon can be expected to result in network congestion and message loss – problems that should have been removed by adding the fast backbone medium in the first place.

Therefore, a closer look on the issue of congestion control appears worthwile. This paper takes a generic approach that is not limited to the restricted case of high speed backbones and low speed fieldbus segments. First, general observations are made. Then, the situation with respect to KNX is discussed as a case in point. Since this protocol originally did not include a communication medium with comparatively high bandwidth at all and is a very light protocol in general, the issues mentioned can be expected to be especially pronounced. Finally, a number of possible remedies are suggested. Although they address the shortcomings identified for KNX, many of them are applicable in a broader context.

II. Preconditions and requirements

Keeping a potentially faster sender from overrunning a (slower) receiver has been a frequent requirement since the classic days of computing; this task is usually referred to as *flow control*. On the other hand, controlling traffic entry into a communication network to avoid the condition that little or no useful communication is happening due to the amount of lost or dropped messages burdening the network (also known as "congestive collapse") is usually known as *congestion control*. For the purposes of this paper, we shall follow this definition:

flow control throttles a sender according to the processing speed of the receiver; congestion control does so with the goal of avoiding oversubscription of network resources.

It is important to note that throttling the sender is, on principle, only possible at all if the (application) semantics of the data to be transmitted allow their rate to be adjusted. For non-realtime data such as configuration messages or program downloads or most kinds of diagnostic information, this is the case; the allowable delay is only limited by economic concerns. Realtime data however become obsolete after a certain time that depends on the application. For some kinds of data, especially absolute value reports or modifications ("set valve position to 60%", "the current room temperature is ..."), this expiration time may not be static and a certain band between a minimum and a desired update rate may exist. In certain cases, higher delay and jitter are acceptable as long as the timestamp of an event is preserved (e.g., for trending; this obviously requires synchronized network time).

Besides timeliness, message delivery guarantees are another parameter of interest that follows from the requirements made by the application (error semantics). While some messages will neither tolerate loss nor duplication (chiefly, this applies to those containing value offsets, e.g., "increase meter count by 10" or "toggle status"), duplication is not a problem with idempotent commands and status reports (absolute value modification, value report). If these are sent periodically (instead of solely in response to a value change), even message loss is acceptable as long as the minimum update rate is met.

If the same message is sent to multiple receivers (e.g., a controller and a visualization panel), each one may have different timeliness requirements. In this case, the slowest receiver determines the maximum message rate if message loss can not be tolerated and reliable transmission is required.

On its way from the sender to the receiver, a message may be delayed or lost due to insufficient resources in several places. First, the segment that the sender is connected to may experience an overload condition. Second, another segment along the route to the receiver may be congested, causing a router queue to overflow (or the message to become stale while still in the queue). Third, the receiver may be too busy to process the message. While the sender is implicitly aware of the first condition, adapting to the second and third requires explicit feedback via the network to close the loop.

Overload conditions on a segment can occur no matter what its throughput rating is (just as traffic jams are common on highways). Fairness and relative prioritisation (at least of realtime vs. non-realtime traffic) should be ensured at every traffic entry point into a segment (i.e., end devices and routers).

Congestion on one segment should not needlessly impede traffic on neighbouring segments that still have headroom capacity. Neither should a busy device. If application semantics allow reducing the message transmission rate, such measures should be limited to traffic inbound to the obstructed segment or busy device.

Even the best congestion control cannot help if the network is simply underprovisioned to handle the desired application. In case of data with a limited time of validity (realtime data), considerations should be made whether the network can carry the worst case message load. If error-free transmission is assumed, this is a straightforward calculation as long as maximum data entry rates (and minimum intervals) are defined for all end devices (obviously, without such limitations, no guarantees can be made at all). Both time triggered and event triggered designs are capable of adequate performance if dimensioned properly. Event triggered designs may exhibit more jitter and may be more difficult to analyze (more complex queuing will usually be involved), but are more flexible in allocating network capacity to where it is needed.

A network designed to withstand theoretical worst case load conditions is seldom economical. Event triggered designs allow designing for the average case; they will still deliver loads that are above average provided that these are compensated by below-average conditions somewhere close-by in time or space. While this works reasonably often, it means relying on stochastics and thus can also go wrong (especially if the assumptions made during design were not reasonable or no longer hold since conditions have changed). Therefore, it should be possible to diagnose overload conditions. Without support for such diagnostics, the presence of overload conditions can only be guessed from external observation of system malfunction. Ideally, it should also be reported if the message loss was tolerable from a semantic point of view or if it was compensated by retry mechanisms. Also, message travel and expiration times would be of interest to judge the quality of service or take measures improving network performance (e.g., by rearranging a router queue or eliminating stale messages from it).

III. Message semantics and congestion handling in KNX

KNX supports multiple communication media. While the twisted pair medium (TP1, around 1990) and the 230 V powerline communication option (PL110, around 1996) originate from its EIB heritage, the radio frequency medium (RF) was added more recently. Around 1997, "EIB.net" was specified as a first fast backbone medium (based on IEEE 802.3 Ethernet, more specifically ISO/IEC 8802-2 LLC). However, it did not meet with success and was withdrawn. Around 2004, EIBnet/IP (now KNXnet/IP) was introduced. KNXnet/IP provides tunnelling over IP networks in various separate modes, most importantly a point-to-point mode allowing flow control ("KNXnet/IP Tunnelling") and a point-to-multipoint mode intended for connecting routers via a fast backbone ("KNXnet/IP Routing"). Almost instantly, KNXnet/IP Routing was suggested as a basis for supporting "native IP" devices as well ([10]). In 2008, KNX Association officially announced that communication over IP would be introduced in the near future as a first class KNX medium, and that "KNX IP devices will use the same [...] message format as KNXnet/IP Routing" [11]. In the following, relevant aspects of the KNX protocol will be introduced and discussed.

KNX application layer (AL) services fall into two distinct groups: device configuration/diagnosis and exchange of process values/data. The services supported by a device are specified in its system profile. Device configuration/diagnosis services use either the connection oriented (reliable) or the connectionless (unreliable) point-to-point communication service provided by the transport layer (TL); again, the device profile defines the valid choice. A special case within this first group are the services for device disambiguation ("discovery"), which use the TL broadcast service.

For process communication, KNX follows a producer-consumer communication pattern. The communication endpoints of devices for this purpose are referred to as "group object datapoints" or "group objects". Every group object represents a single input or output value (which may – but most often does not – consist of multiple fields that are transmitted simultaneously). Binding is done by assigning identical "group addresses" to the group objects whose values are to be kept in sync. However, multiple outputs can be bound to a single input, replacing this model of coupled states with generic message channels.

The AL services for process communication consist of an unconfirmed service to announce the update of a shared value (A_GroupValue_Write) and a confirmed service to ask for the current state of a shared value (A_GroupValue_Read). The former is the mainstay of KNX communication. It is not only used for reporting that the value of a point has changed, but also for all kinds of state change or action requests. Although a mode of runtime communication ("LTE", Logical Tag Extended) has been introduced that provides distinction between value change orders ("Write") and reports ("InfoReport"), this mode is entirely separate from standard KNX runtime interworking ("S-Mode") from which such a distinction remains absent.

A_GroupValue_Write is used for messages associated with any kind of error semantics. Although most messages contain absolute point values, very often these messages (e.g., "Stop sunblind movement", "Adjust output level to 20 %FS") are sent upon a change of value (desired state) only. If multiple output group objects are bound to the same input group object – a popular and necessary practice, e.g., for "master light switch" applications –, periodic retransmissions are not possible at all. As for messages carrying value offsets, "toggle state" commands and "meter impulse" reports are not unheard of; with them, unlike with absolute values, inconsistencies introduced by way of message loss or duplication will not rectify themselves over time and cannot be tolerated.

The AL services for process communication exclusively rely on the TL multicast service (which is never used by the AL device configuration/diagnosis services). For all TL services, the data link layer (DL) provides a first level of duplicate removal (via flagging and filtering retransmissions, or via a 3-bit sequence counter on RF). The connection oriented unicast TL service adds an additional sequence count to compensate for message loss and to improve duplicate removal; for all other services, the TL takes no additional measures over what is done by the DL. Neither do the network layer or the AL.

Thus, messages whose semantics do not tolerate loss use an AL service that does not offer reliable delivery. If confirmations are required, they have to be obtained by the user application. Actually, such "end-to-end" status feedback at the application level has the advantage of being able to account for messages that were delivered successfully, but ignored by the application (e.g., due to an interlock, such as a wind alarm blocking sunblind operation despite a command to move). However, integrators often forego this option for performance reasons. Also, the status feedback (itself delivered via an unreliable service) may be lost. Thus, KNX systems more often than not include communication relationships where the loss of a single GroupValue_Write protocol data unit (PDU) will lead to an inconsistent system state resulting in (light or possibly severe) malfunction. Up to now, this has been masked by the fact that messages hardly ever got lost.

A possible motivation for the absence of a reliable multicast service might be that realtime data become obsolete after some time anyway, rendering potential retransmission attempts pointless (and the intention that the multicast service would never be used for non-realtime data). However, as has been detailed, not all messages become obsolete.

Compared to A_GroupValue_Write, A_GroupValue_Read is only seldom used. It does not share the semantic ambiguities of A_GroupValue_Write. However, as it is also sent via multicast, multiple devices could answer a request, even with different values (as a single group address can correspond with multiple datapoints). Thus, the device to answer has to be preselected at configuration time. Adding a further twist, the remote confirmation (i.e., answer) to this service is considered equivalent to a GroupValue_Write PDU by a significant share of stack implementations in use.

No AL process communication service reflects timeliness requirements. However, these can differ significantly. For example, a GroupValue_Read request PDU may need to be answered within seconds (for storing present light levels as a scenario preset) or several minutes (when a visualization software application initializes its process image).

Four message priority levels are supported at the data link layer, which can be regarded as a way to express relative timeliness requirements. These levels appear as service parameters to the application layer user; the standard only loosely regulates their use (TL acknowledgment PDUs are to be sent with the highest priority; in general, the priority of configuration/management procedures is above process communication). The standard engineering tool software (ETS) allows the project engineer to choose one of the lower three priority levels per group object.

Message ordering is not guaranteed to be preserved. However, before the advent of KNXnet/IP Routing, message ordering was always kept due to the acyclic network topology (and routers operating in a strict FIFO manner). If KNXnet/IP Routing is used, reordering may happen on the IP medium.

With connection-oriented TL services, a new message is sent only after the remote TL acknowledgment PDU (ACK)

has been received. While this procedure already causes the sender to slow down according to the network round trip delay, delays in upper layers and the user application (management server) are not (necessarily) factored in. However, most of the device AL configuration/diagnosis services either expect a reply or an end-to-end confirmation (readback) is obtained by the management server. Since these are always waited for (only for a few management procedures, static timeouts are defined), end-to-end flow control is ensured. In high device or traffic load situations, the conservatively, but statically defined TL retransmission timeout (3 s) or connection timeout (6 s) may expire while messages are still in transit or being processed, causing unproductive traffic.

The TP1 data link layer also offers a kind of flow control mechanism: "If the request frame received is correct but the remote Data Link Layer doesn't have resources to process it, the remote Data Link Layer shall send a BUSY character" [6]. Since this mechanism is limited to the local segment (which is appropriate for a data link layer mechanism), it cannot be used for end-to-end flow control. The PL110 data link layer specification does not include this mechanism.

As for congestion control, it is effectively impossible for a sender to obtain feedback about the conditions on the path to the receiver (except via end-to-end acknowledgments). The data link layer BUSY acknowledgment being referred to as BUSY/FULL in one place in the DL specification suggests that a coupler (KNX router) could use it to announce that its routing queue is full; the network layer specification contains nothing in this regard. KNXnet/IP Routing provides the ROUTING_LOST message, which is to be sent on the IP network when messages are dropped from the routing queue.

Both of these mechanisms only provide information about the next hop. Moreover, neither allows deducing which destination is affected by congestion ("doesn't have resources to process it" can as well mean that the destination address check could not be performed; ROUTING_LOST does not volunteer information regarding the messages that were lost save a total count). Thus, no selective reaction is possible; senders can only blindly throttle their overall transmission rate.

Even if a sender knew that the path to a particular destination is congested, it would not be able to selectively relieve this path from traffic, since an end device has no information about the correlation between group addresses and network topology – a message to a different group may take exactly the same path. Even couplers only see part of this information: the subset contained in their routing table.

The handling of load peaks on the local segment is medium specific. On TP1, devices with a sufficiently high left-right (MSB-LSB) swapped device address may experience starvation due to the bitwise arbitration mechanism used (for messages within a group of priority levels). The TP1 data link layer specification calls for measures that guarantee access fairness by introducing an additional waiting period, but without specifying any details. PL110 specifies a randomized slotted offset mechanism (with devices picking one of seven slots after the minimum waiting period at random, the choice being influenced by the message priority) to reduce the risk of collisions (which cannot be detected on this medium). On RF, where the situation is similar, the interframe time is also to be extended with a random delay; while time ranges are given, no clear specifications are made. In particular, no correlation between message priority and wait time is suggested (the required interframe time for RF retransmitters, however, is well below that of end devices). KNXnet/IP Routing does not include any mechanisms to allow prioritisation or ensure access fairness. This is hardly a problem when the IP segment is populated only with IP/TP couplers; however, it may easily become an issue when end devices use a form of EIBnet/IP Routing as their native mode of communication, as buffers in network switches may overflow.

Offline, configuration-time measures for preventing network overload and/or congestion are rare. Load calculations are usually performed by rule of thumb. Only HVAC function blocks for LTE devices specify maximum and typical message generation rates. The network load generated by these devices can thus be roughly specified as a characteristic number and serve as a guideline for network planning.

Virtually no standard mechanisms exist for diagnosing overload conditions. In addition to the ROUTING_LOST message, KNXnet/IP Routing specifies two mandatory counters (diagnostic properties) that hold the number of packets dropped from either routing queue. No such mechanism is specified for couplers in general or for messages that an end device could not place on its local segment in time. Also, no provisions exist for associating time with message transmissions.

IV. Suggestions for improvement

A number of suggestions for KNX can be derived from these observations. All of them can be implemented independently of each other and in a backward compatible manner. Most do not introduce new PDUs or traffic overhead; some do not require touching the communication protocol at all.

As an "offline" measure towards congestion prevention, function blocks should be amended to specify typical and minimum message generation intervals required by the application they are a part of (as already done for the LTE HVAC function blocks). These figures can then be used to estimate network load (possibly even automatically as a feature in ETS).

At runtime ("online"), the likelihood of congestions (in remote segments) can be reduced if devices that emit a burst of messages to different destinations (e.g., parallel program downloads, activating a complex scenario or initializing the process image of a management application) randomize the order of these outgoing messages.

To improve the robustness of the system in high load conditions, communication endpoints should be bound and datapoint types (DPTs) should be chosen to match the unreliable nature of the process communication services. Absolute datapoint values transmitted periodically (which does not preclude additional transmissions on demand) will allow a more graceful degradation of system performance and easier recovery. Where

appropriate, end-to-end status feedback should be included to ensure consistency.

If message loss could be tolerated, interesting possibilities would open up. For instance, TCP uses packet loss as a congestion indicator and adapts accordingly. The same would be possible on KNX not only for management connections, but also for periodic process traffic that tolerates variations of its period length. Sources could maintain and send along a sequence count per destination address, and receivers could monitor it to obtain information about the loss rate per source – which they periodically report back to them. However, it is a fact that EIB systems were and KNX S-Mode systems will be designed according to the reliability one has come to expect from the communication mechanisms in use, without virtually any regard to the semantics that are actually guaranteed. Therefore, from a purely pragmatic point of view, efforts must be made to keep the probability of the connectionless, unreliable TL services failing as low as possible.

Specifically for the use of IP as a first-class medium, the handling of situations where a larger number of devices attempt to access the medium at the same time should be improved. In particular, such a situation can occur when a state change of a group of devices is triggered by a single Group-Value_Write PDU and these devices are configured to report status updates. On IP over switched Ethernet, the network switch takes the role of the common medium. Since KNX medium definitions seek to emulate the inherent broadcast nature of the original twisted-pair bus medium, IP traffic patterns result that are unusual in the IT world. Although the performance of many current network switches allows them to route traffic between their ports at wirespeed, this becomes increasingly impossible given the multicast pattern of KNX traffic once enough KNX IP devices are connected. Once a switch queue overflows, packets are dropped, resulting in what amounts to a collision on the shared medium. It cannot be assumed that devices can detect such a loss condition – which resembles the situation on open media.

It may therefore be advisable to look at how such situations are handled on other KNX media (not at least since adopting similar procedures would have the added benefit of being consistent with tradition). Actually, every KNX medium in use to date (with the natural exception of RF in case of unidirectional devices) specifies busy detection and random wait times once the medium is free. Something similar could easily be implemented for KNX IP, having the random waiting period start with the last KNX IP frame received. Depending on the operating system, precise timing of IP packets may not be possible. Yet even then, such a procedure should reduce the likelihood of "collisions" and improve access fairness. In addition, message priorities can be allowed for by factoring them into the calculation of the waiting period.

The algorithm for determining these wait times should be very clearly specified to avoid the risk of a race between device manufacturers for the most aggressive interpretation (in order to make their devices perform better in peak load situations). This does not only apply to IP, but also to other KNX media.

Quality of service information is important for allowing graceful degradation and improving robustness in congestion situations. Therefore, KNX message priorities need to be chosen wisely. The KNX specification ties configuration/diagnosis traffic to the highest priority levels. However, such traffic is non-realtime and very often uses connection-oriented transport that tolerates packet loss. From this perspective, one would rather expect it to use the lowest priority level. Possible reasons behind the surprising choice made in the KNX specification may be the assumption that such traffic will not be present during normal operation anyhow and the consideration that if it is, it should be able to break its way through a congested network to correct the overload issue by reconfiguration.

Setting message priorities only has an effect if messages with higher priority also receive privileged treatment. Strangely, message priorities are entirely ignored on KNX RF. Also, one would expect routers to insert messages into their queues according to priority. However, this simple measure apparently is not mentioned in the specification. Although it would influence the delivery order of messages, KNXnet/IP Routing can cause the same today. The KNX specification should provide clarification on the topic of reordering to avoid false assumptions by developers and integrators.

To address the issue of multicast destinations having different quality of service requirements (aside from simply picking the higher priority for both), the destinations can be assigned separate group addresses, each with different message priorities. Of course, this is only useful if the path to the low priority destinations is not a subset of the path to the high priority destinations. Since the KNX standard only allows a single TSAP (TL service access point, i.e., group address) to be associated for outgoing traffic with any ASAP (AL SAP, i.e., group object), this would require support by the device application by providing additional ("shadow") group objects.

Without doubt, the producer-consumer communication pattern (which, per design, does not include feedback from the consumer – or anywhere along the path between producer and consumer, for that matter) and a great disparity in processing and communication speeds do not go well together.

A solution to this problem has to introduce a means of feedback. Basically, messages need to be defined that allow devices and couplers to announce overload conditions. In couplers, overload will usually caused be insufficient buffer size (or a time threshold for placing messages in the routing queue on the remote interface being exceeded) rather than insufficient processing resources. Ideally, these messages will include a suggested retry time when the overload condition is expected to have disappeared. Preferably, they should be sent before message loss has actually occurred. If messages have already been lost, this should be announced differently to enable the sender to react appropriately. These announcements may have to be sent with elevated priority.

The announcements must specify traffic to which destinations is to be held back by potential senders so the overload condition can be reduced. For individually addressed traffic, this is easy; the device address, or, for routers, the affected

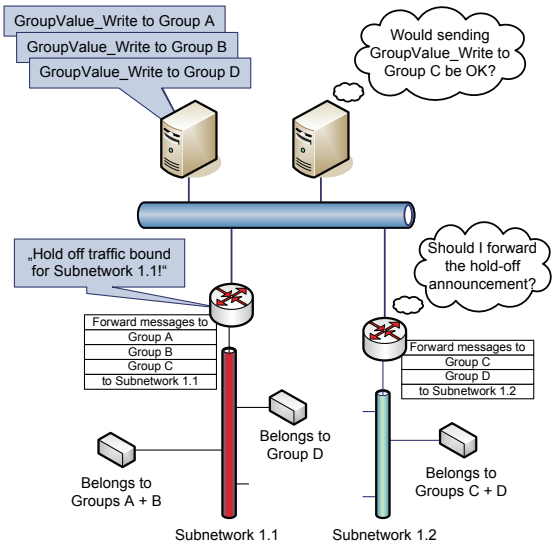

Fig. 1. Congestion avoidance: Issues

subnetwork (zone/line) number suffices. For group traffic, this is only possible if senders know which devices are included in any particular group (or, the other way round, which group addresses will create traffic inbound to a particular device and thus subnetwork). This information may be preloaded into the devices during configuration via ETS; it may be included in the overload announcement message; or it may be dynamically obtained by reading out device address tables and coupler filter tables at runtime.

While sending the information along with every announcement would be less complex, transmitting the entire mapping tables at startup (with low priority multicast) and on request (for devices joining later) seems preferable as it would reduce the traffic overhead to a negligible amount. Of course, holding the tables in memory requires additional device resources. Devices that cannot store the tables (or have not had a chance to receive them yet) can still indiscriminately reduce traffic generation upon receipt of an overload announcement.

Rules are required for the propagation of these announcements to other segments. Broadcasting them would be the easiest way to catch all potential traffic sources, but could easily swamp low rate segments and low performance devices, making matters worse. Rather, couplers should store announcements for their validity period and only pass them on when a frame with a held-off destination comes in via their other interface during this period (possibly also if such frames have come across in a certain time period before receiving the overload announcement, to account for temporal locality). Fig. 1 illustrates some of the issues discussed.

While mechanisms for performance optimization help to increase efficiency and quality of service, they do not guarantee a particular quality of service. Underprovisioning, legacy devices that do not implement the new mechanisms or configuration errors may still lead to packet loss or unacceptable delays of realtime messages. Therefore, diagnostic mechanisms are required. At the very least, counts of messages dropped by couplers and messages that could not be sent by end devices in time due to congestion on the local segment should be available in a standardized, medium independent fashion. Timestamping of realtime traffic would be even more helpful. Standard DPTs for time synchronization exist; the available precision would be sufficient as long as synchronization is performed during periods of minimal network load. However, adding timestamps to process data in a backward compatible way would incur a significant amount of additional traffic.

Management connection clients usually are PC based and have ample free resources. To optimize network usage, they could easily monitor round trip times (using a sliding window approach) and adapt their transmission rate accordingly (as it is done by the TCP flow control mechanism). Of course, the KNX TL timeouts need to be taken into account; while the client can easily adapt them too, the server will still use the static standard values. They could however also be made dynamic, possibly set by the client via new control messages to relieve the server from the necessary calculations.

V. Outlook

Clearly, the effectiveness of the proposed mechanisms remains to be examined, possibly by way of simulation. Appropriate protocol extensions need to be implemented. The findings need to be set into the context of the relevant literature with respect to flow and congestion control and shared medium emulation (specifically, but not limited to Ethernet and IP networks), delivery semantics, quality of service and queuing theory in general. Also, the analysis should be extended to other relevant protocol standards such as CNP and BACnet.

The strict hierarchic, depth-limited structure of KNX systems may allow additional assumptions and approaches (especially in the increasingly typical case of a two level topology with a high-speed backbone and low-speed fieldbus segments).

After this paper had been written and reviewed, KNX Association published version 2.0 of the KNX specifications, including a draft proposal for IP as a first class KNX medium (KNX IP, as previously announced). The findings in this paper were forwarded to KNX Association for consideration.

References

[1] ANSI/ASHRAE Std. 135, *BACnet – A Data Communication Protocol for Building Automation and Control Networks*, 1995–2008.
[2] ISO 16484-5, *Building automation and control systems (BACS) – Part 5: Data communication protocol*, 2007.
[3] EN 14908, *Open data communication in building automation, controls and building management – Control network protocol*, 2005.
[4] ANSI/EIA/CEA Std. 852, *Tunneling Component Network Protocols Over Internet Protocol Channels*, 2002.
[5] LonMark International, http://www.lonmark.org
[6] Konnex Association, *KNX Specifications, Version 1.1*, 2004.
[7] Konnex Association, *KNX Specifications, Version 2.0*, 2009.
[8] EN 50090, *Home and Building Electronic Systems (HBES)*, 1994–2007.
[9] EN 13321-2, *Open data communication in building automation, controls and building management – Home and building electronic systems – Part 2: KNXnet/IP Communication*, 2005.
[10] F. Heiny et al., "Virtual KNX/EIB devices in IP networks", *KNX Scientific Conference 2004*, Deggendorf (available from www.knx.org).
[11] "KNX IP – a new class of KNX devices", *KNX Journal 1/2008*, pp. 12–13 (available from www.knx.org).