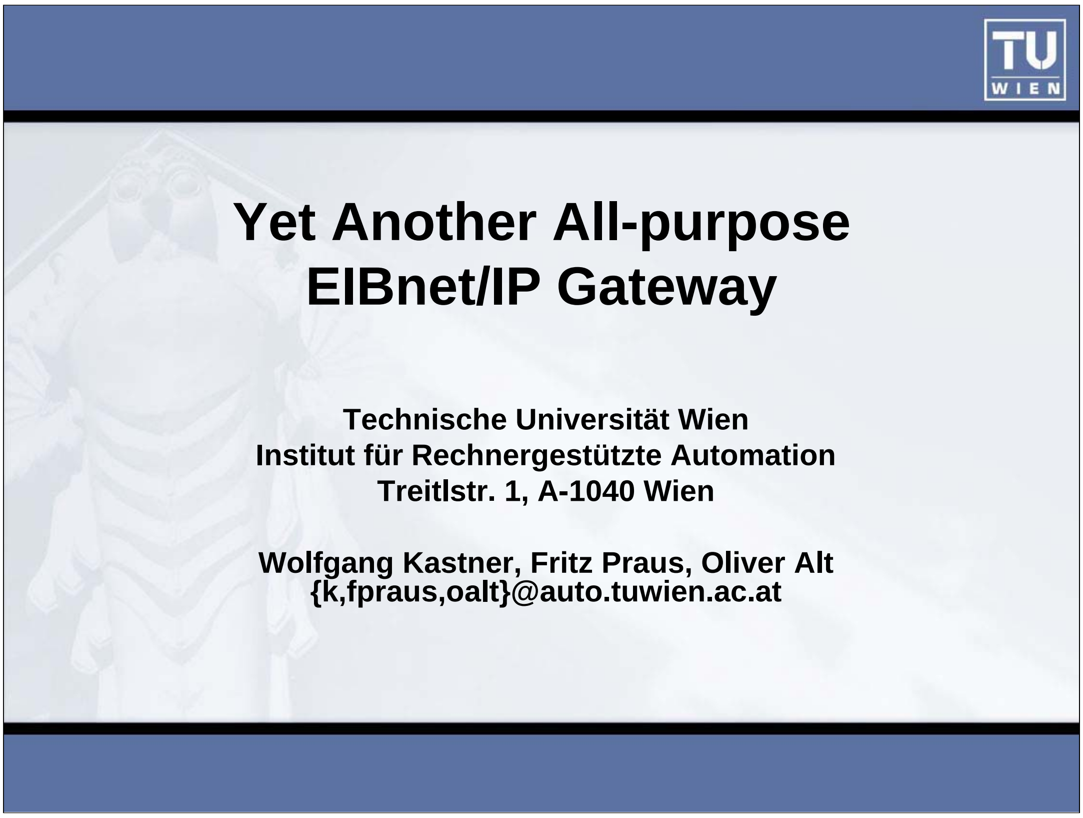


Yet Another All-purpose EIBnet/IP Gateway

Technische Universität Wien
Institut für Rechnergestützte Automation
Treitlstr. 1, A-1040 Wien

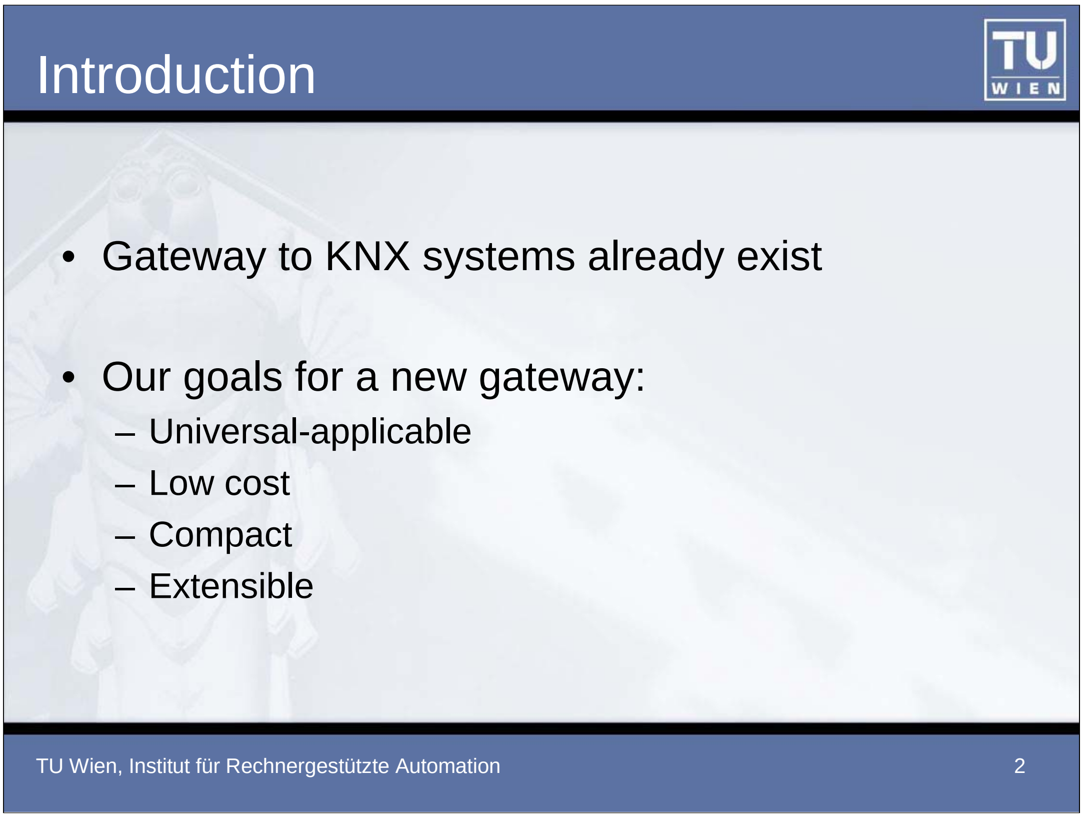
Wolfgang Kastner, Fritz Praus, Oliver Alt
{k,fpraus,oalt}@auto.tuwien.ac.at

Abstract

Currently, we are designing a universal-applicable gateway to KNX systems. The hardware is based on an Fujitsu 16bit micro-controller with 24 MHz, 24Kbyte RAM, 384Kbyte flash memory, full USB support, 4 UARTs, 3 channels for I^{2}C bus communication and an external bus interface. Our gateway will support EIBnet/IP (tunneling and routing) with an embedded web-server facilitating HEAD, GET and POST requests and CGI scripts.

The gateway is intended as universal platform for ongoing development concerning integration of other fieldbus and multimedia bus systems to KNX, as well as gateway for remote IP services. In addition, it serves as a base for work in progress regarding the development of a set-top box for integration into the OSGi environment and plug and play networks.

The paper presents the design of the hardware interfaces and interesting parts of the software implementation.



Introduction

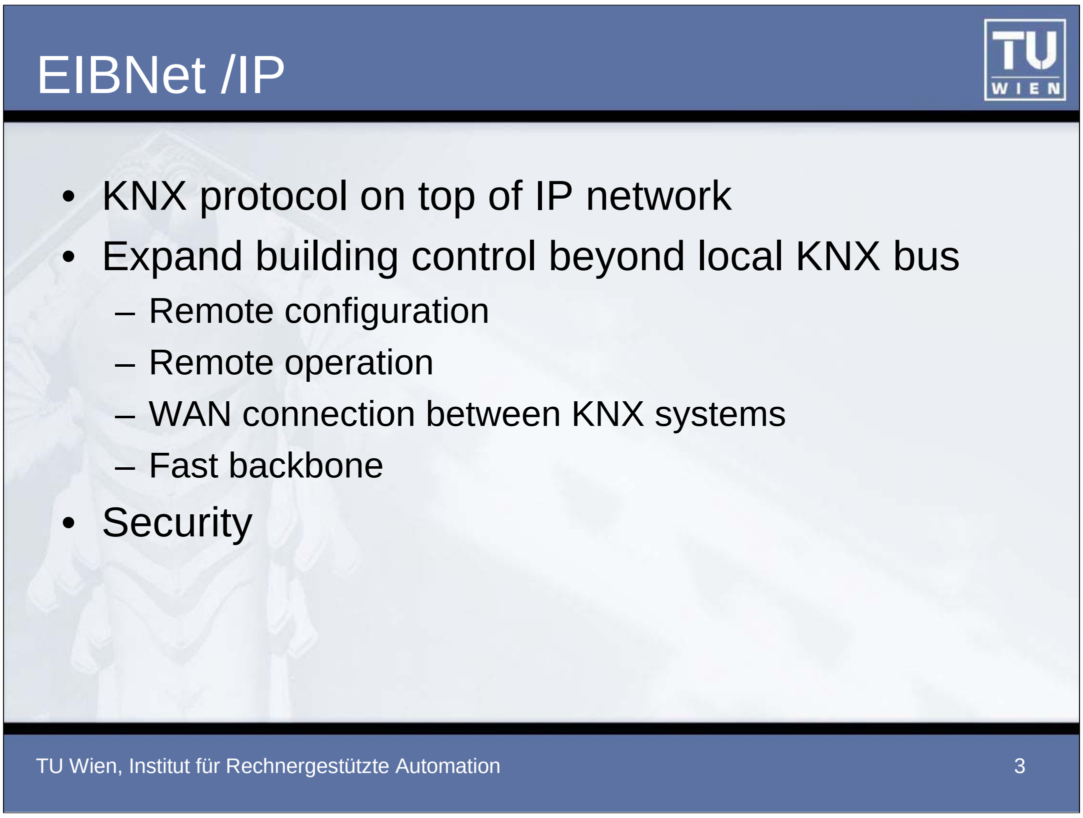
- Gateway to KNX systems already exist
- Our goals for a new gateway:
  - Universal-applicable
  - Low cost
  - Compact
  - Extensible

TU Wien, Institut für Rechnergestützte Automation 2

Without question, a lot of EIB gateways are already on the market.

The gateway presented in this paper is based on the work of Oliver Alt, that had a connection to Ethernet on one side and a coupling to EIB on the other side.

The primary goal for our new gateway is, that it should be be universal-applicable. The main purpose is to serve as a basis for further extensions regarding our work in the scope of home and building automation (i.e., plug and play facilities, integration into OSGi environments, coupling to other networks with regard to security issues, setup of set-top boxes, access point for BASys). Furthermore it should be compact, low cost and able to be built from scratch with cheap “components of the shelf”. Most important, no additional hardware or drivers shall be needed.



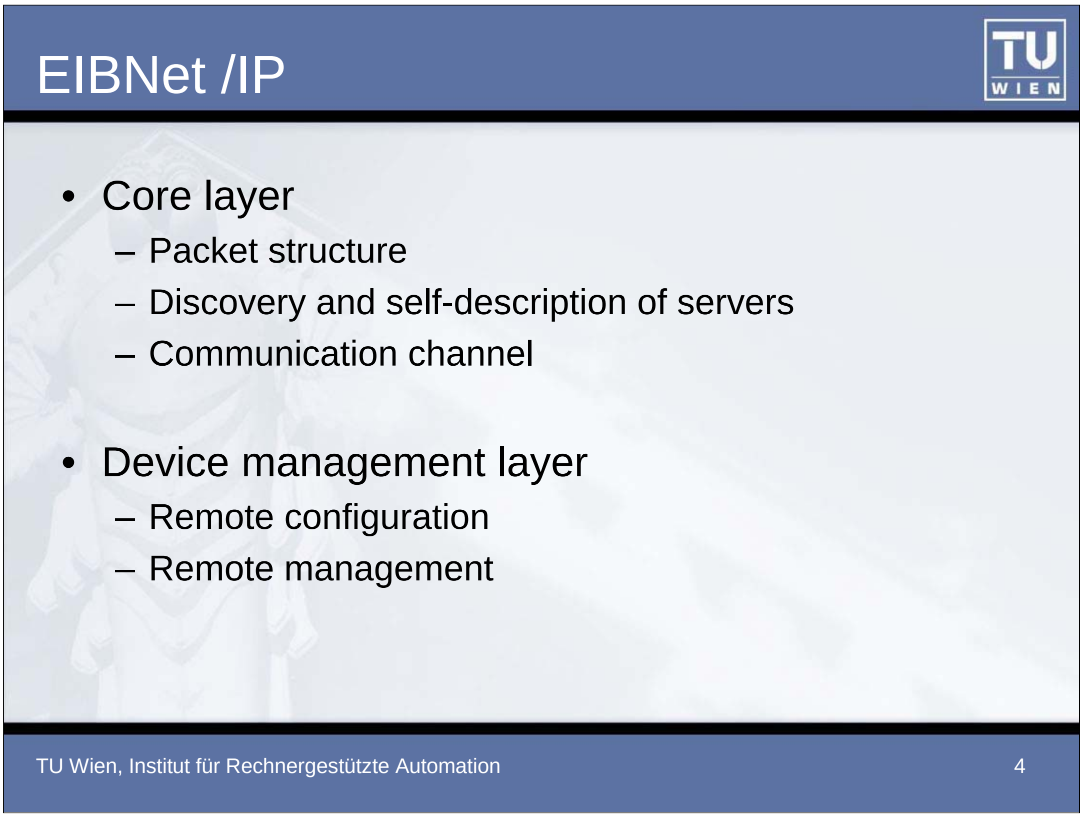
EIBNet /IP

- KNX protocol on top of IP network
- Expand building control beyond local KNX bus
  - Remote configuration
  - Remote operation
  - WAN connection between KNX systems
  - Fast backbone
- Security

TU Wien, Institut für Rechnergestützte Automation 3

EIBnet/IP describes transportation of KNX telegrams on top of IP networks, which are, due to their widespread deployment, an ideal fast backbone. The main purpose of EIBNet/IP is to expand building control beyond the local KNX bus. EIBNet/IP is transparent for KNX devices. Remote configuration and operation is possible via EIBNet/IP servers.

Until now, for KNX security was of minor concern, as any breach of security requires physical access to the network wires, which is nearly impossible in a building. When using an existing data network, however, several security threats need to be considered. These are, for example, eavesdropping, modification or deletion of messages and denial of service attacks. As EIBNet/IP does not provide any security measures, the underlying network has to be secured by, for instance, using virtual private networks, local intranet only or using authentication for opening tunnelling or remote logging connections.



EIBNet /IP

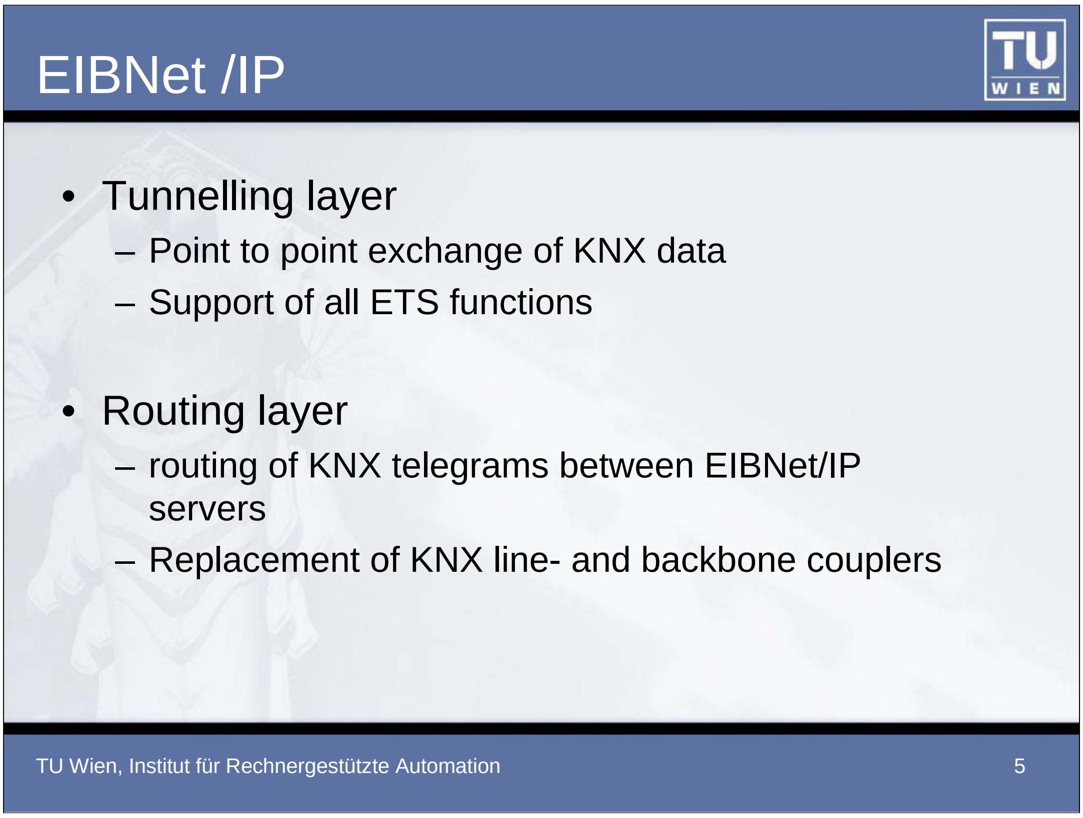
- Core layer
  - Packet structure
  - Discovery and self-description of servers
  - Communication channel
- Device management layer
  - Remote configuration
  - Remote management

TU Wien, Institut für Rechnergestützte Automation 4

Currently there are four layers specified in EIBNet/IP. Every EIBNet/IP device needs to implement at least the core layer.

The core specification describes the packet structure. Moreover, it is responsible for discovery and for self-description of an EIBNet/IP server. Therefore, every server provides a well-known discovery endpoint to the IP network. For establishing and maintaining a communication channel, a server shall at least support one control endpoint and one data endpoint.

The device management layer is responsible for configuration and management of the device. This can be done via the IP network or the KNX network. The layer is responsible for interpreting the cEMI telegrams carried by EIBNet/IP telegrams.



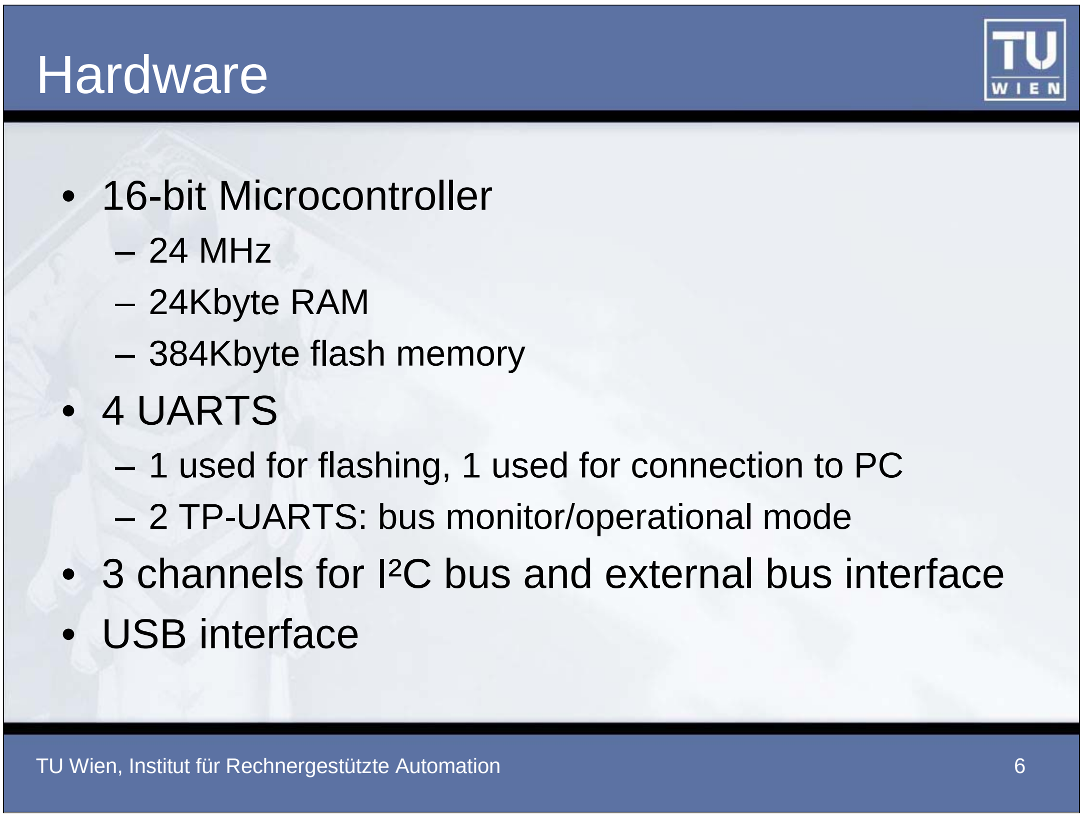
EIBNet /IP

- Tunnelling layer
  - Point to point exchange of KNX data
  - Support of all ETS functions
- Routing layer
  - routing of KNX telegrams between EIBNet/IP servers
  - Replacement of KNX line- and backbone couplers

TU Wien, Institut für Rechnergestützte Automation 5

The tunnelling protocol describes the point to point exchange of KNX data over the IP network for configuration and diagnostics of devices on the KNX network. The tunnelling client sends management telegrams, contained in an IP telegram to the tunnelling server, which passes the data to the KNX network. It has to be mentioned, that the protocol does not address timing issues caused by the IP network. All ETS functions are supported by tunnelling devices.

Routing is a point-to-multipoint protocol for routing between KNX devices over the IP network. EIBNet/IP routers send UDP/IP multicast messages to other EIBNet/IP routers on the same IP network, that in turn filter the messages according to their destination address or group address and eventually pass them to the KNX layer. EIBNet/IP routers can replace traditional KNX line and backbone couplers.



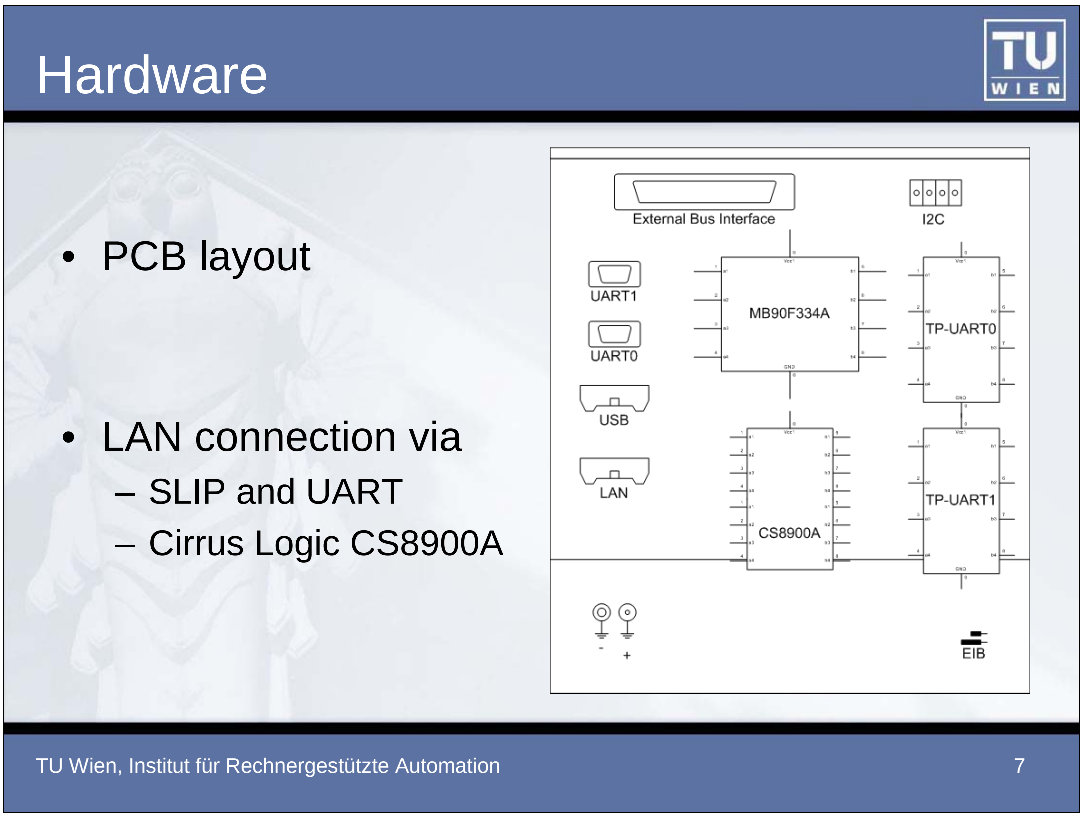
Hardware

- 16-bit Microcontroller
  - 24 MHz
  - 24Kbyte RAM
  - 384Kbyte flash memory
- 4 UARTS
  - 1 used for flashing, 1 used for connection to PC
  - 2 TP-UARTS: bus monitor/operational mode
- 3 channels for I^{2}C bus and external bus interface
- USB interface

TU Wien, Institut für Rechnergestützte Automation 6

The central part of our hardware is a Fujitsu Siemens MB90F334A 16-bit microcontroller. It features 24Kbyte RAM, 384Kbyte flash memory and is operated at 24 MHz. Furthermore, it offers 4 UARTS, 3 channels for I^{2}C bus, an external bus interface and an USB interface.

The controller has enough performance and RAM to support multiple connections and multiple services like EIBNet/IP and a web server. To support bus monitoring as well as “normal” operation mode on the KNX side, two TP-UARTS are connected. Moreover, the two TP-UARTS can be connected to different KNX networks, enabling the device to act as a comprehensive router. The remaining two UARTS are used for flashing and connection to the PC. Connection to the PC will also be possible over the USB interface conformable to the KNX/USB specification. The external bus interface of the microcontroller is routed to circuit points on the side of the board.



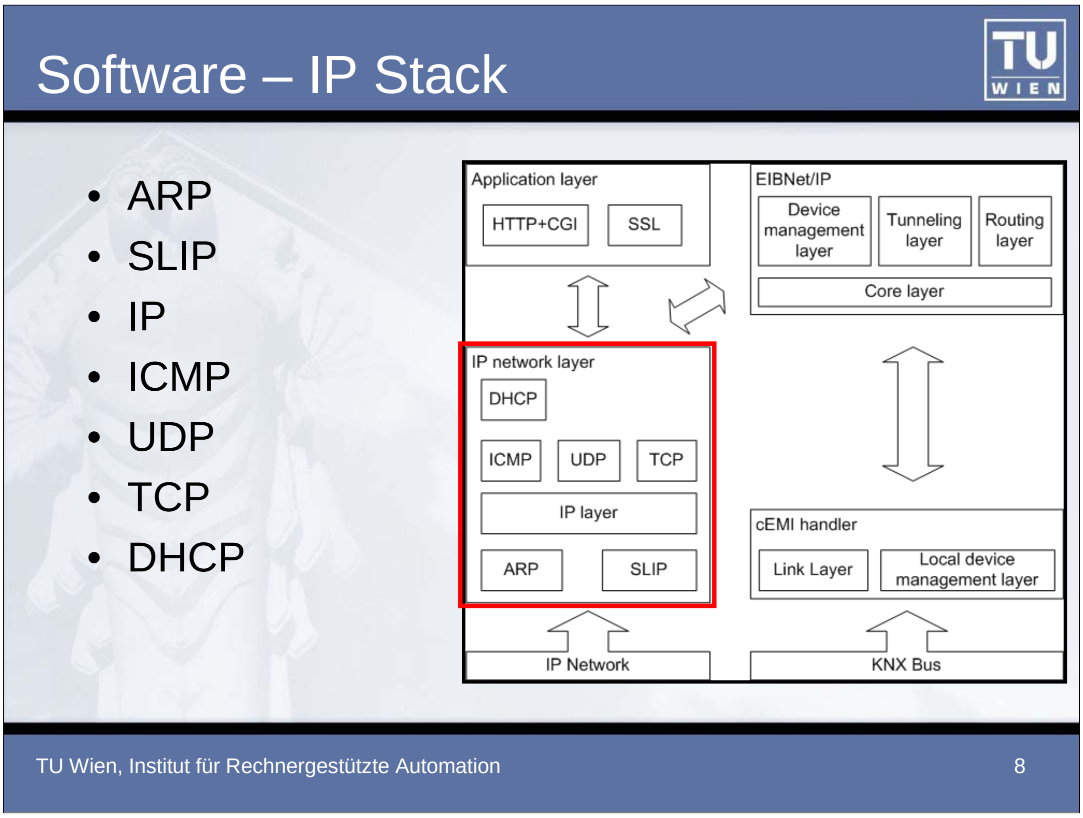
Hardware

- PCB layout
- LAN connection via
  - SLIP and UART
  - Cirrus Logic CS8900A

TU Wien, Institut für Rechnergestützte Automation 7

The connection to the IP network is realised in two ways. Firstly, there is the possibility to connect the gateway over SLIP (a protocol to transport IP packages over a serial line). It is rather simple and used for testing our prototype. Secondly, the gateway can be connected over a Cirrus Logic CS8900A Ethernet chip. This is a low cost Ethernet LAN controller, featuring an ISA Bus interface, 10MBit connection speed and integrated RAM. It is connected to the external bus interface of the microcontroller, which is similar to the ISA Bus.

The outline of the circuit board is shown in the slide above.



Software – IP Stack

- ARP
- SLIP
- IP
- ICMP
- UDP
- TCP
- DHCP

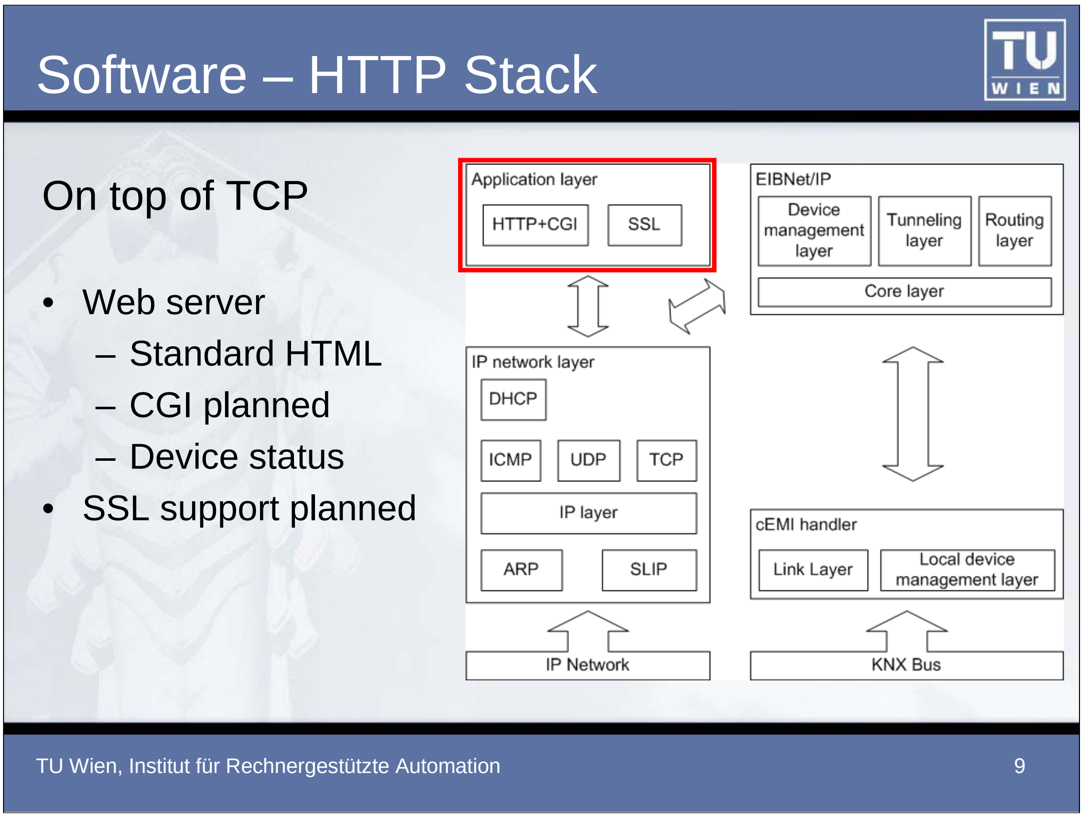
TU Wien, Institut für Rechnergestützte Automation 8

The firmware of the microcontroller is implemented in various layers and handlers. First, the SLIP was implemented. By “emulating” a modem device, a connection is established with the host PC and IP packages can be transferred. When using the CS8900A, ARP is implemented to map IP to MAC addresses. For obtaining an IP address, DHCP was realised. If no DHCP server is present, the device has a hard coded IP.

Upon the reception of an IP package, it is handled by the so called “IP Manager”, that checks the destination address and eventually passes the packet to the ICMP, UDP or TCP handler.

For testing purposes, ICMP was partially implemented. The gateway responds to ICMP echo packets. For the moment, the TCP implementation is in a primary phase. The device only supports one connection to the destination port 80 (HTTP). The UDP implementation is finished. It was fully implemented and is currently used for EIBNet/IP part of the gateway.

It has to be mentioned, that currently there exists one receiving queue for three packets from the IP network. The queue is a ring buffer, where new packages are added interrupt driven and addressing is handled by two pointers. Due to the fact that only one packet can be processed at a given time, there only exists one sending “queue” for IP packages.



Software – HTTP Stack

On top of TCP

- Web server
  - Standard HTML
  - CGI planned
  - Device status
- SSL support planned

Application layer

HTTP+CGI

SSL

EIBNet/IP

Device management layer

Tunneling layer

Routing layer

Core layer

IP network layer

DHCP

ICMP

UDP

TCP

IP layer

ARP

SLIP

cEMI handler

Link Layer

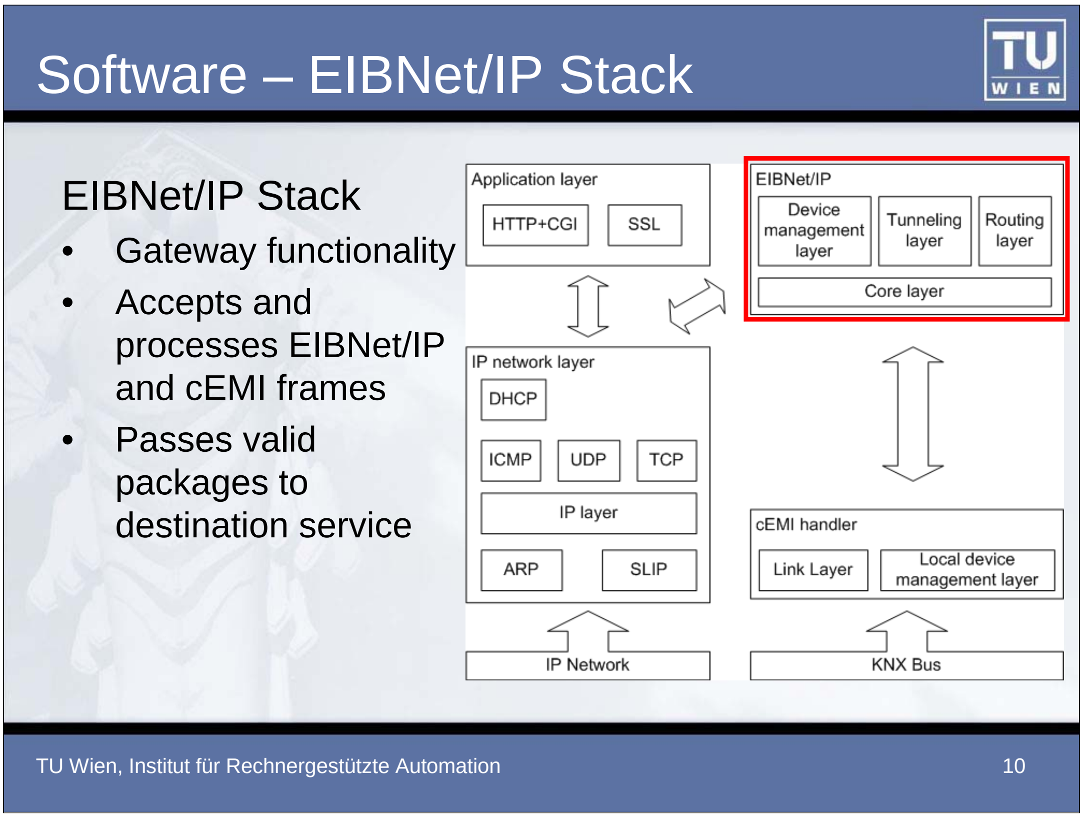
Local device management layer

IP Network

KNX Bus

TU Wien, Institut für Rechnergestützte Automation 9

For the moment, HTTP (port 80) is the only implemented TCP service. The current device status is returned in a formatted html page. As a further extension, the integration of a scripting language like CGI is planned. Furthermore SSL will be implemented for establishing a secure connection to the gateway over an insecure IP network.



Software – EIBNet/IP Stack

EIBNet/IP Stack

- Gateway functionality
- Accepts and processes EIBNet/IP and cEMI frames
- Passes valid packages to destination service

Application layer

HTTP+CGI

SSL

EIBNet/IP

Device management layer

Tunneling layer

Routing layer

Core layer

IP network layer

DHCP

ICMP

UDP

TCP

IP layer

ARP

SLIP

cEMI handler

Link Layer

Local device management layer

IP Network

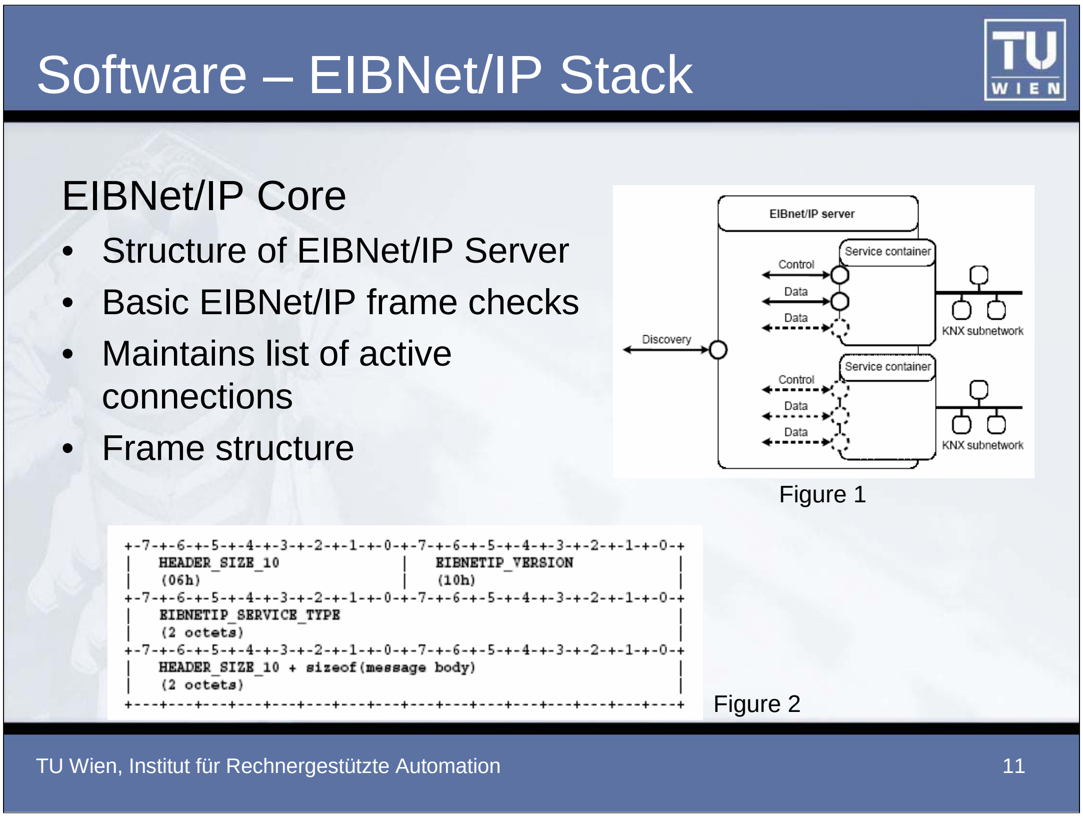
KNX Bus

TU Wien, Institut für Rechnergestützte Automation 10

The EIBNet/IP stack is positioned on top of the IP network layer and on top of the cEMI handler and acts as a gateway between the two networks. On the one hand it accepts EIBNet/IP frames from UDP/TCP level and on the other hand it accepts cEMI frames from the cEMI handler. It processes the received frames according to their message types and passes valid packages to the destination service.

The EIBNet/IP stack on this gateway consists of 4 layers:

- Core layer,
- Device Management layer,
- Tunneling layer, and
- Routing layer



Software – EIBNet/IP Stack

EIBNet/IP Core

- Structure of EIBNet/IP Server
- Basic EIBNet/IP frame checks
- Maintains list of active connections
- Frame structure

Figure 1

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   EIBNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   EIBNETIP_SERVICE_TYPE                                       |
|   (2 octets)                                                  |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + sizeof(message body)                       |
|   (2 octets)                                                  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

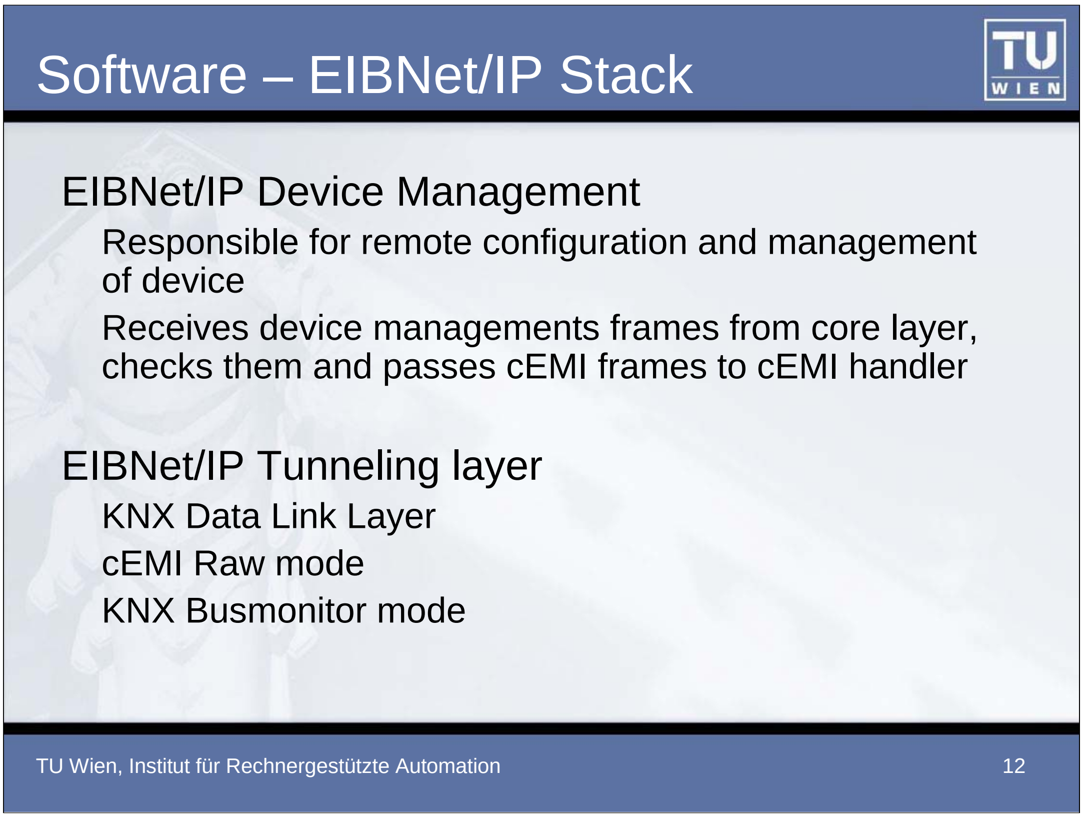
Figure 2

TU Wien, Institut für Rechnergestützte Automation 11

The core layer is the basic layer of EIBNet/IP. As mentioned before, it is responsible for discovery and self-description of the device and for configuring, establishing, and maintaining a communication channel with an EIBNet/IP client. Figure 1 explains the basic structure of the EIBNet/IP server. The gateway provides one well known discovery endpoint and one or more service containers. If the gateway is connected to two different KNX networks, it provides two different service containers with according control and data endpoints.

An EIBNet/IP telegram consists of a common EIBNet/IP header and variable length EIBNet/IP body. The structure of the header is shown in Figure 2. The first octet is the header size, which for the moment is fixed to six octets. The second octet stands for the EIBNet/IP version, which is currently fixed to 1.0. The following two octets describe the EIBNet/IP service type, according to which the telegram is passed to the destination layer. The last two octets of the header are the size of header and size of message body.

The EIBNet/IP core layer interprets the EIBNet/IP header and provides basic frame checks for incoming telegrams. If header size, message version and message size are correct, the telegram is passed to the specific handler. Furthermore, this layer is responsible for maintaining a list of currently active connections. Due to the fact, that UDP, which does not provide any services for correct message order or message delivery, is used for transportation, the layer also has to handle these missing services. A sequence counter for incoming and outgoing messages is maintained by the layer.



Software – EIBNet/IP Stack

EIBNet/IP Device Management

- Responsible for remote configuration and management of device
- Receives device managements frames from core layer, checks them and passes cEMI frames to cEMI handler

EIBNet/IP Tunneling layer

- KNX Data Link Layer
- cEMI Raw mode
- KNX Busmonitor mode

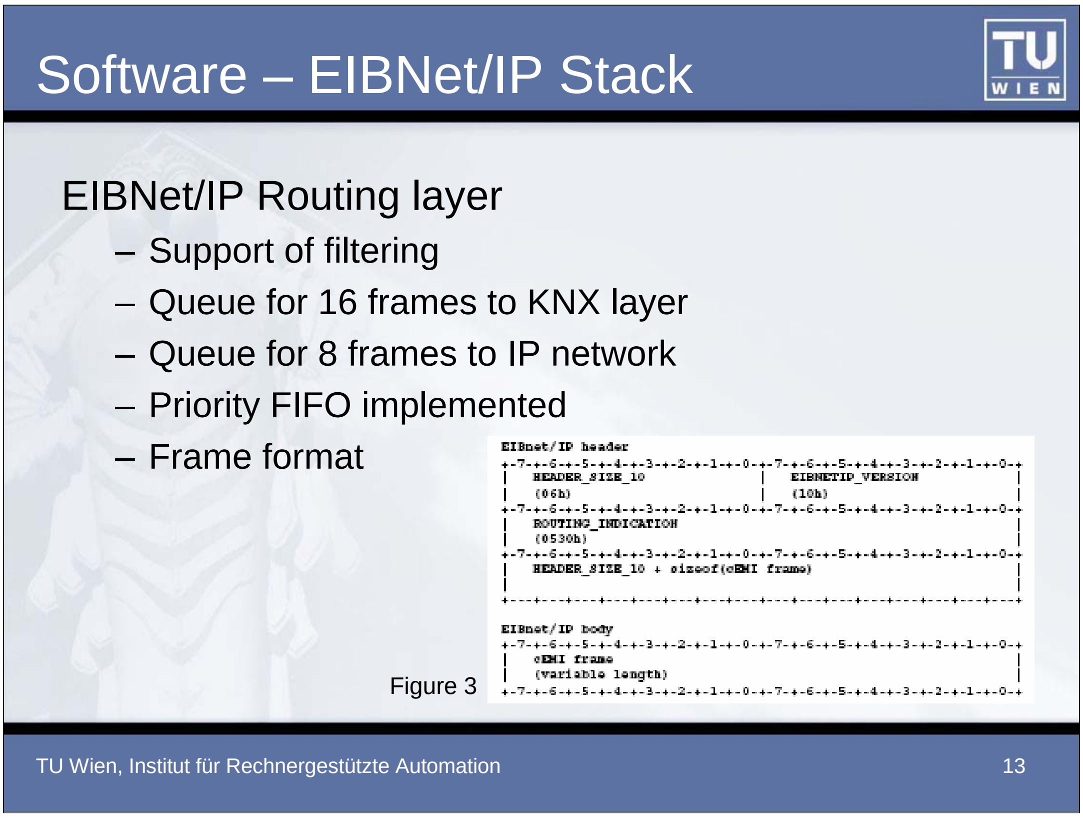
TU Wien, Institut für Rechnergestützte Automation 12

The EIBNet/IP Device Management layer is responsible for remote configuration and management of the device. It accepts EIBNet/IP packets carrying cEMI frames, checks them and passes the cEMI frames to the cEMI handler, which then processes the packets. The advantage of configuring the device over EIBNet/IP is the support of larger data structures.

The gateway supports tunnelling in all three specified modes. Upon establishing a communication channel for tunnelling, the gateway assigns each tunnelling connection a KNX individual address. After establishing the communication channel, an EIBNet/IP client can send `TUNNELING_REQUEST` frames containing an EIB/KNX telegram.

In “tunnelling on KNX Data Link Layer” mode, only packets from the KNX bus with the destination address equal to the assigned address are passed to the EIBNet/IP client. Furthermore, all telegrams on KNX point-to-multipoint addressing are forwarded to the client.

In “cEMI Raw mode” and “KNX Busmonitor mode” all received packets from the KNX network are passed to the EIBNet/IP client.



Software – EIBNet/IP Stack

EIBNet/IP Routing layer

- Support of filtering
- Queue for 16 frames to KNX layer
- Queue for 8 frames to IP network
- Priority FIFO implemented
- Frame format

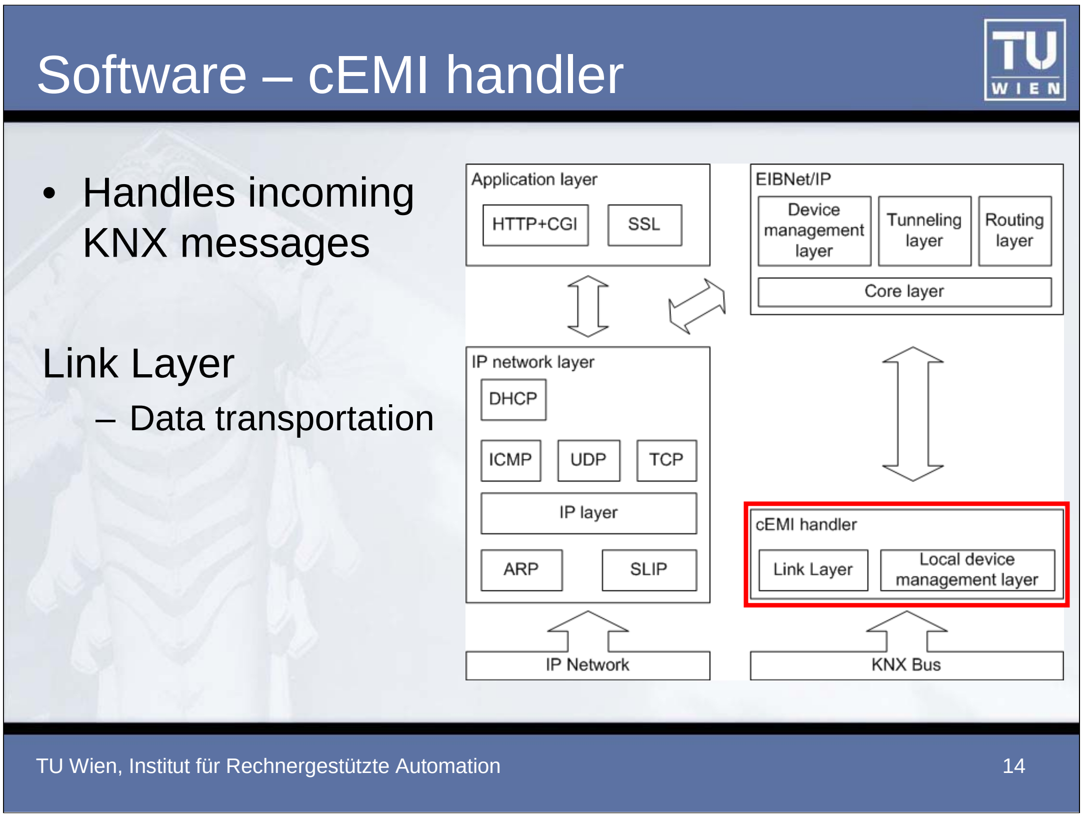
```
EIBnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   EIBNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   ROUTING_INDICATION                                          |
|   (0530h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + sizeof(cEMI frame)                         |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

EIBnet/IP body
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   cEMI frame                                                  |
|   (variable length)                                           |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
```

Figure 3

TU Wien, Institut für Rechnergestützte Automation 13

The gateway supports filtering of telegrams according to their destination address. It is able to handle messages from the IP network at 10MBit/s. The KNX bus has a much slower speed and thus messages to KNX layer have to be queued. The gateway features a sending queue of 16 frames to KNX layer and 8 frames to the IP network. The queue, however, can overflow resulting in message loss. The gateway then increments a counter and sends a routing lost message, which can be logged by a central supervising entity. As a forwarding rule, priority FIFO is implemented: telegrams with the highest KNX priority are routed firstly, even if they are queued later than other telegrams.

Figure 3 describes the simple frame format. The EIBNet/IP frame is encapsulated in an UDP/IP packet and consists of the common header and the attached cEMI frame.



Software – cEMI handler

- Handles incoming KNX messages

Link Layer

  – Data transportation

Application layer
HTTP+CGI
SSL

EIBNet/IP
Device management layer
Tunneling layer
Routing layer
Core layer

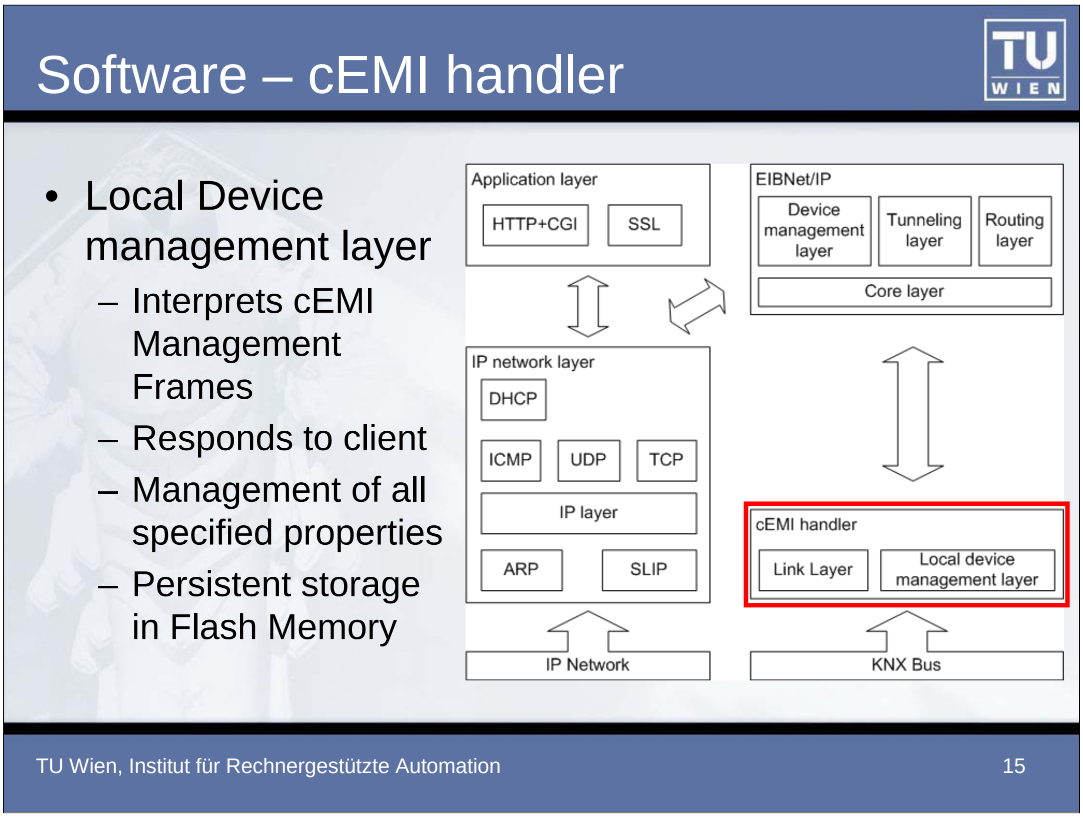
IP network layer
DHCP
ICMP
UDP
TCP
IP layer
ARP
SLIP

cEMI handler
Link Layer
Local device management layer

IP Network
KNX Bus

TU Wien, Institut für Rechnergestützte Automation 14

The cEMI handler is responsible for handling incoming cEMI telegrams from the KNX bus and from the EIBNet/IP layer. Telegrams from the KNX side are saved in the receiving queue after an interrupt request from the TPUART. Depending on the gateway state and the message type, the cEMI message is handled differently. If the gateway is in routing or tunnelling mode, packets with the corresponding destination addresses are passed from Link Layer to the EIBNet/IP layer, which in turn generates EIBNet/IP frames and passes them to the IP layer. The IP layer then tunnels/routes the package to the IP network. If the EIBNet/IP handler is “shutdown”, only messages equal to the device’s individual address are processed by the cEMI handler. In this case, the Link Layer is “out of function” and only device management via the KNX bus is possible.



Software – cEMI handler

- Local Device management layer
  - Interprets cEMI Management Frames
  - Responds to client
  - Management of all specified properties
  - Persistent storage in Flash Memory

Application layer
HTTP+CGI
SSL

EIBNet/IP
Device management layer
Tunneling layer
Routing layer
Core layer

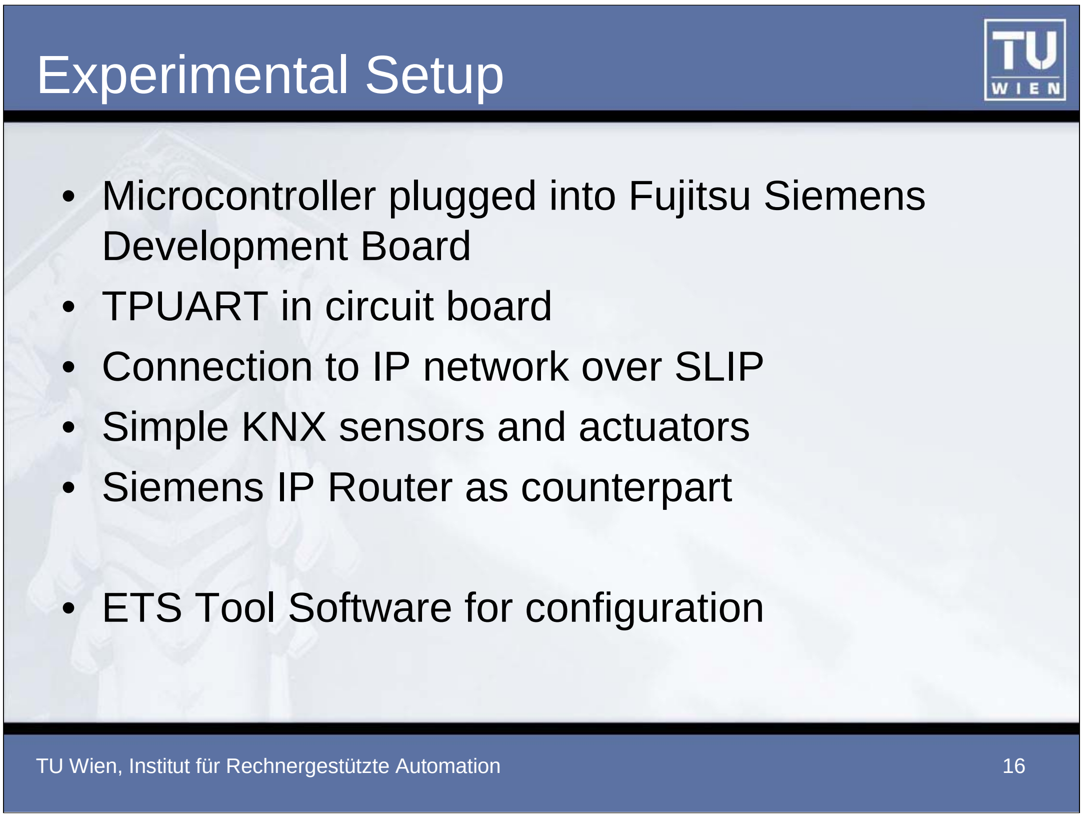
IP network layer
DHCP
ICMP
UDP
TCP
IP layer
ARP
SLIP

cEMI handler
Link Layer
Local device management layer

IP Network
KNX Bus

TU Wien, Institut für Rechnergestützte Automation 15

The local device management layer interprets cEMI management frames, received by the KNX bus or by the EIBNet/IP layer. All gateway properties, like for example KNX individual address, device name, …, can be configured. The layer is also responsible for generating the correct responses to the management server. For persistent storage, all data is kept in flash memory.



Experimental Setup

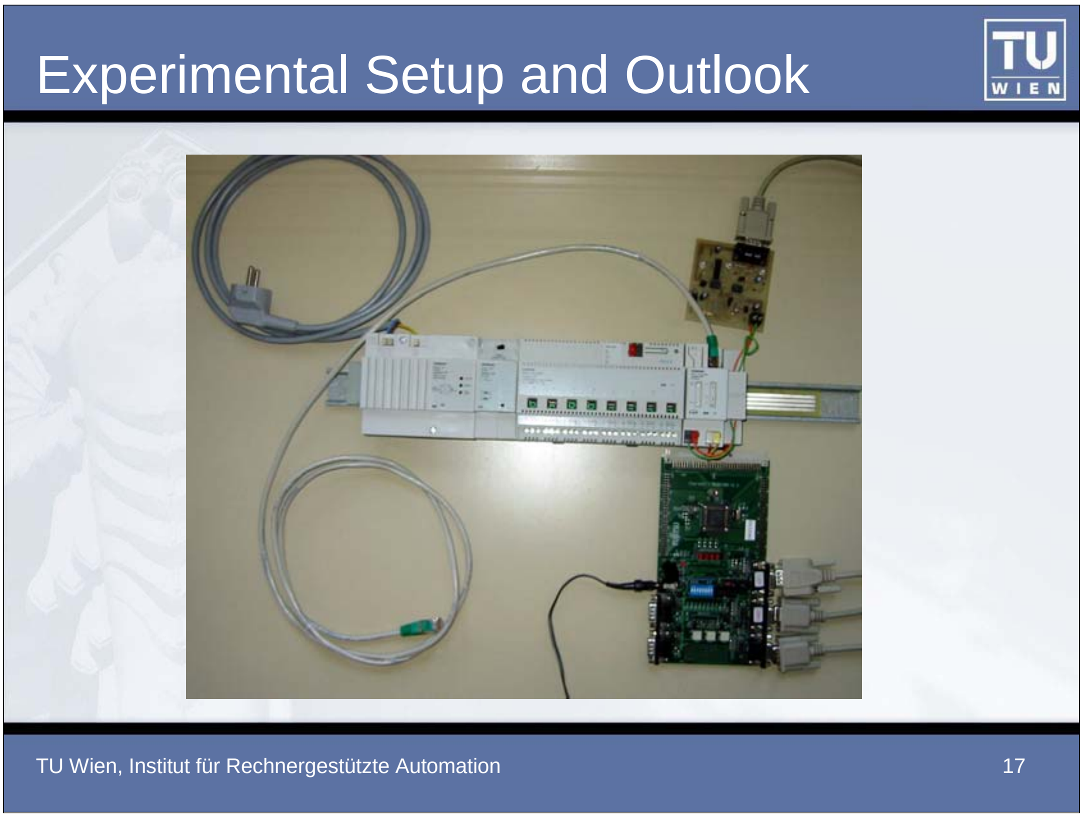
- Microcontroller plugged into Fujitsu Siemens Development Board
- TPUART in circuit board
- Connection to IP network over SLIP
- Simple KNX sensors and actuators
- Siemens IP Router as counterpart

- ETS Tool Software for configuration

TU Wien, Institut für Rechnergestützte Automation 16

The realisation of the final board is “in progress”. For development and testing purposes the microcontroller is plugged into a Fujitsu Siemens development board, where all essential connectors can reached. Furthermore, the development board features a reset button and easy mode switching for flashing and operational mode of the MCU. The TPUART and the required components are plugged into a circuit board, too. It is connected to a UART of the MCU by a serial cable. The necessary drivers for the TPUART are implemented in the firmware of the microcontroller. Connection to IP network is realised over a SLIP connection.

For testing purpose, the experimental setup consists of simple actuators and push buttons, as well as an Siemens IP Router that acts as counterpart for IP connections. For configuration and testing the ETS was used. This setup allows demonstration of the device management, tunnelling and routing capabilities of the gateway.



Experimental Setup and Outlook

TU Wien, Institut für Rechnergestützte Automation 17

The above picture shows the experimental setup with all the included components and cables.

The next step in development of the gateway is to design the exact routing layout of the printed circuit board and have the PCB manufactured and fitted with components. Then the current firmware - that means the low level hardware routines – have to be migrated to the new microcontroller. Furthermore, the integrated web server is going to be extended.