Public

Brief overview of connectivity and what to do with it

A: design and integrate all 8 layers and think out-of-band

Tom Hoogenboom, System Engineering, ASML

V2 | 2-jun2015 | Drachten

Background and purpose

Connectivity, after shelter, food and sleep, seems to be a major focus of humankind.

So we connect up everything – for better or worse.

In our ‘systems’ world, we find connectivity has costs, in terms of design work, complexity and running expenses.

In this presentation we take a look at connectivity, starting with the age-old ISO 7-layer model.

Then we’ll examine an list of functions, sometimes forgotten about, that must be on every designers agenda.

This should help to understand ‘layer 8’: who pays for it and why?

Disclaimer: the views presented are those of the author and not necessarily those of ASML.

Ref: Interconnections, second edition, Radia Perlman, page 1

Summary

Connectivity is about layering, protocols and topology

The SECS stack (1982/1995) sets an example for ‘machine’ connectivity

10 sets of functions should be part of any connectivity design

A reliable design requires a single point of control

So what is layer 8?

Summary

Connectivity is about layering, protocols and topology

The SECS stack (1982/1995) sets an example for ‘machine’ connectivity

10 sets of functions should be part of any connectivity design

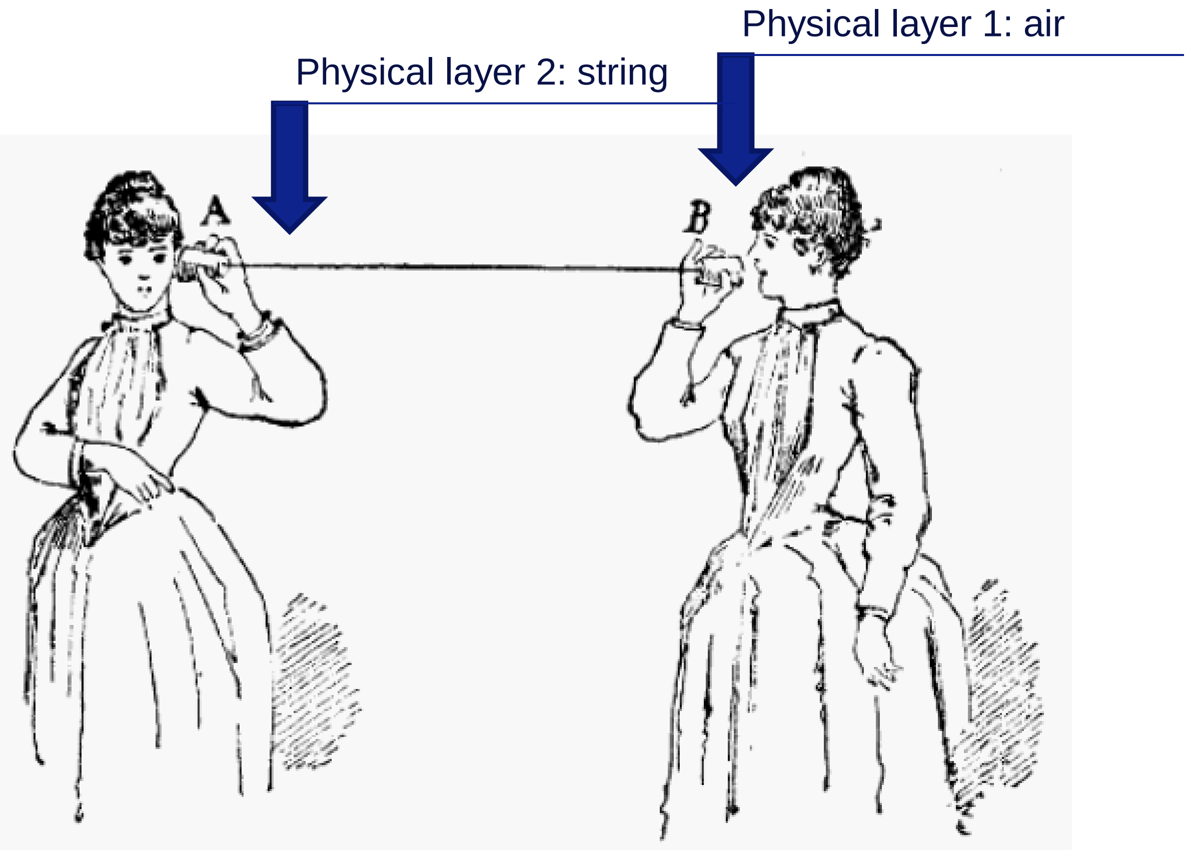
A reliable design requires a single point of control

So what is layer 8?

Connectivity is about layering, protocols and topology

Layering?

Example:



Physical layer 1: air

Physical layer 2: string

Fig. 76. Trådtelefon.

Source: wikipedia

Connectivity is about layering, protocols and topology

Protocols?

Protocol layer 1: Agree to use this method (out of band)

Protocol layer 2: 'over' / 'over & out'

Example:

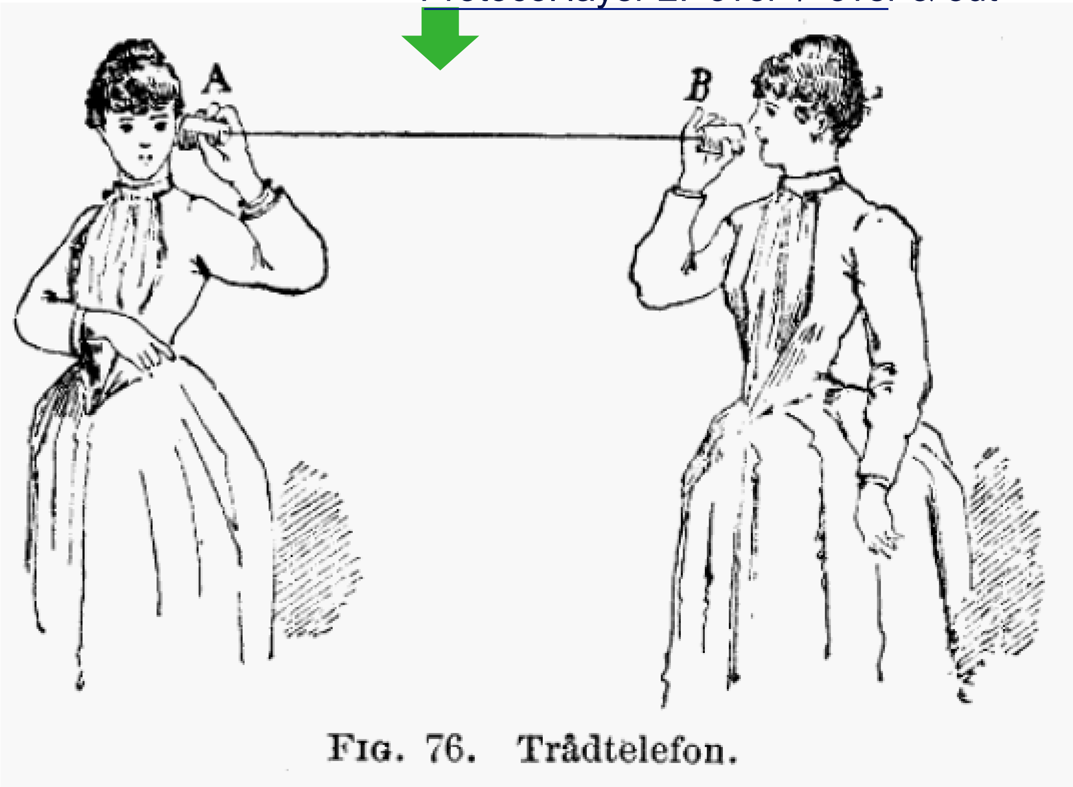


FIG. 76. Trådtelefon.

Source: wikipedia

Connectivity is about layering, protocols and topology

Topology?

Example:

Topology: client server (multipoint)

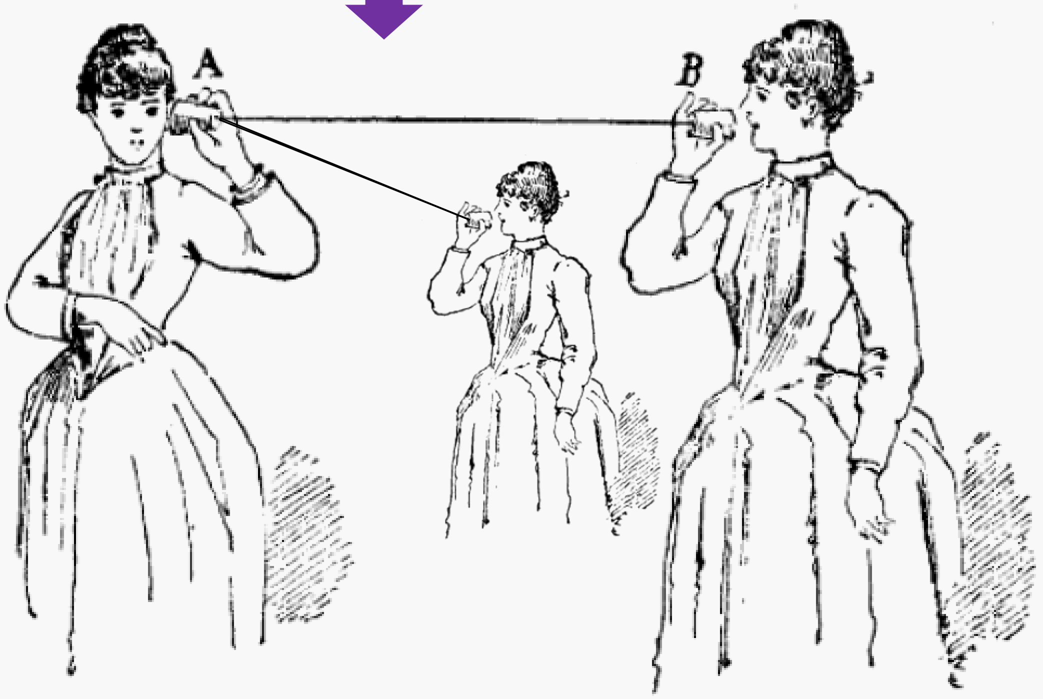


Fig. 76. Trådtelefon.

Source: wikipedia

Connectivity is about layering, protocols and topology

Another example:

Calling Pietje:

- “Pietje there?”
- “位，你好?” (wei, ni hao?)
- “Do you speak English?”
- …

Connectivity is about layering, protocols and topology

example:

Calling Pietje:

- “Pietje there?”
- “位，你好?” (wei, ni hao?)
- “Do you speak English?”
- …

How many layers?

Simple call, at least 3 layers

The phone connection (microphone ↔ loudspeaker, 2-way)

2 people talking to each other

Negotiating a common language

Simple call, at least 3 protocols

The phone connection (microphone ↔ loudspeaker, 2-way)

GSM, 3G, ….

2 people talking to each other

Dutch phone etiquette: start with a question
Chinese phone etiquette: start with 位 (wei, so Hello)

Negotiating a common language

No standard – if English does not work,
and caller does not speak Chinese, good luck….

→ need to go ‘out of band’ to resolve this

Layering allows for separation of concerns

In the examples:

- Transport of audio is separated from negotiating:
    - The physical means for the call (string or GSM)
    - The communication language

So parties are free to select ‘any’ combination of layers

- English over string
- Chinese over string
- English over GSM
- Chinese over GSM

However, the ‘string’ transport is less suitable
for a multipoint conversation in any language

Layering allows for protocol selection per layer

Example:

- GSM transport allows two-way communication
    - Decision to listen is governed by a higher-layer protocol

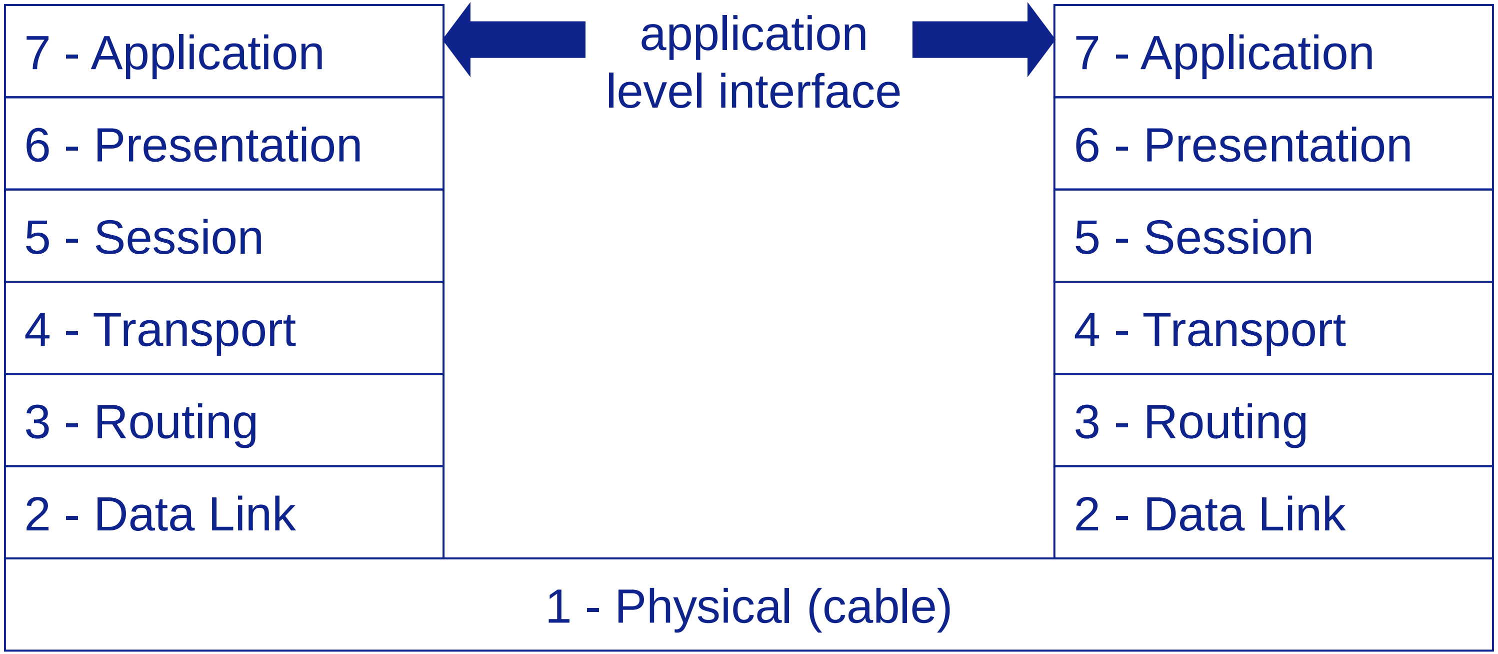
- String transport is one-way, so requires ‘over / over&out’ protocol
    - Higher layer can assume all communication is one-way
    - But communication can breakdown if:
        - ‘over’ message is lost
        - ‘over’ message appears in normal conversation (‘Our relation is over’)
    - So lower layer protocols create restrictions on higher layers.

Both transports are transparent for the ‘negotiate language’ protocol

‘over’ is an example of an ‘out-of-band’ message..

Q: how would this work if there were a language called ‘over’?

ISO-7 sets the standard for layering



7 - Application

6 - Presentation

5 - Session

4 - Transport

3 - Routing

2 - Data Link

application level interface

7 - Application

6 - Presentation

5 - Session

4 - Transport

3 - Routing

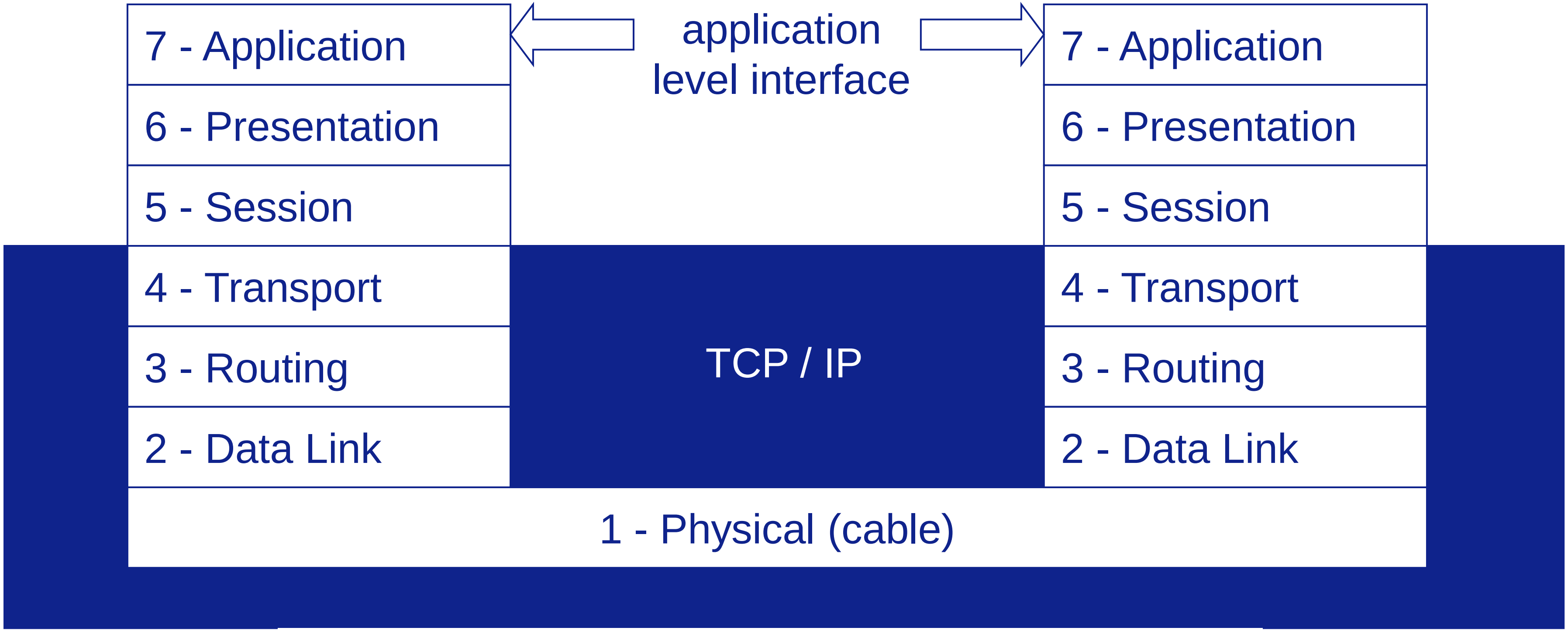
2 - Data Link

1 - Physical (cable)

The ISO 7-layer communication model

routing == iso: network

Today protocols at all layers are mostly TCP/IP



7 - Application

6 - Presentation

5 - Session

4 - Transport

3 - Routing

2 - Data Link

application level interface

TCP / IP

7 - Application

6 - Presentation

5 - Session

4 - Transport

3 - Routing

2 - Data Link

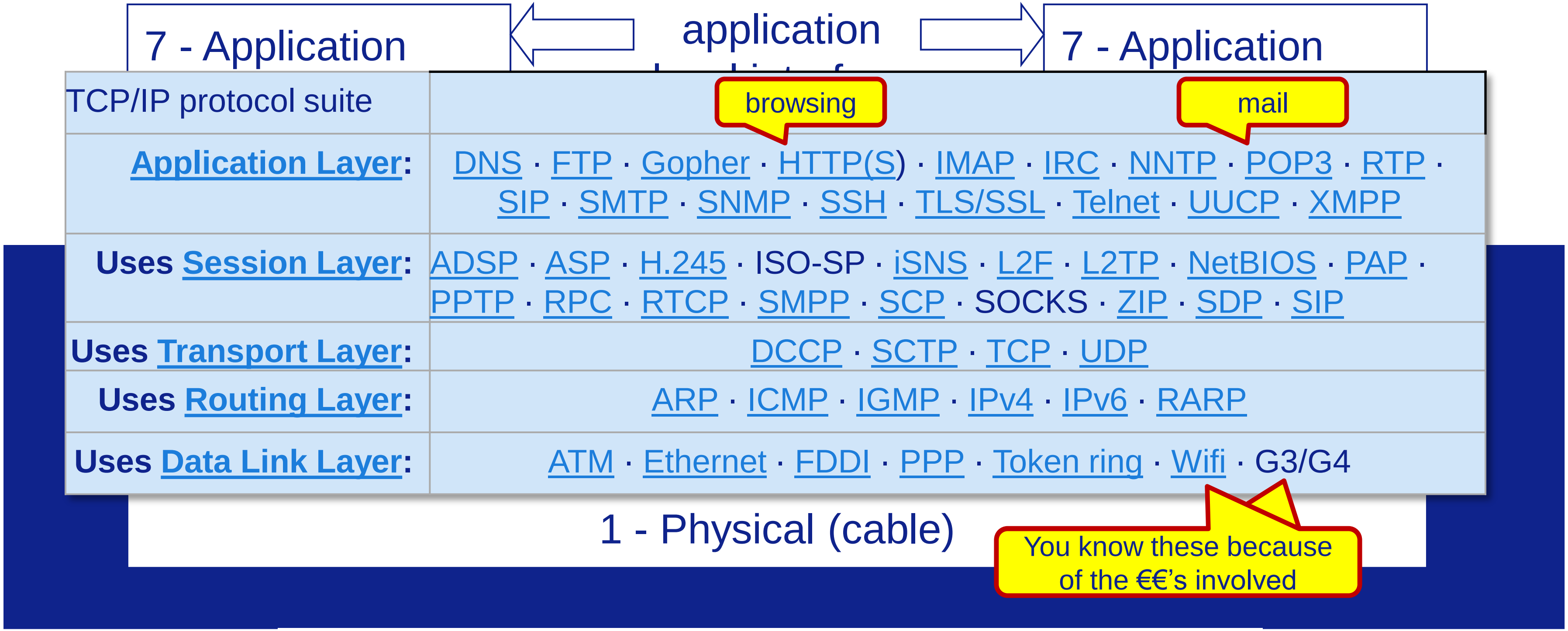
1 - Physical (cable)

The ISO 7-layer communication model

routing == iso: network

Today we use TCP/IP almost without noticing it

Public
Slide 16
2-jun-2015

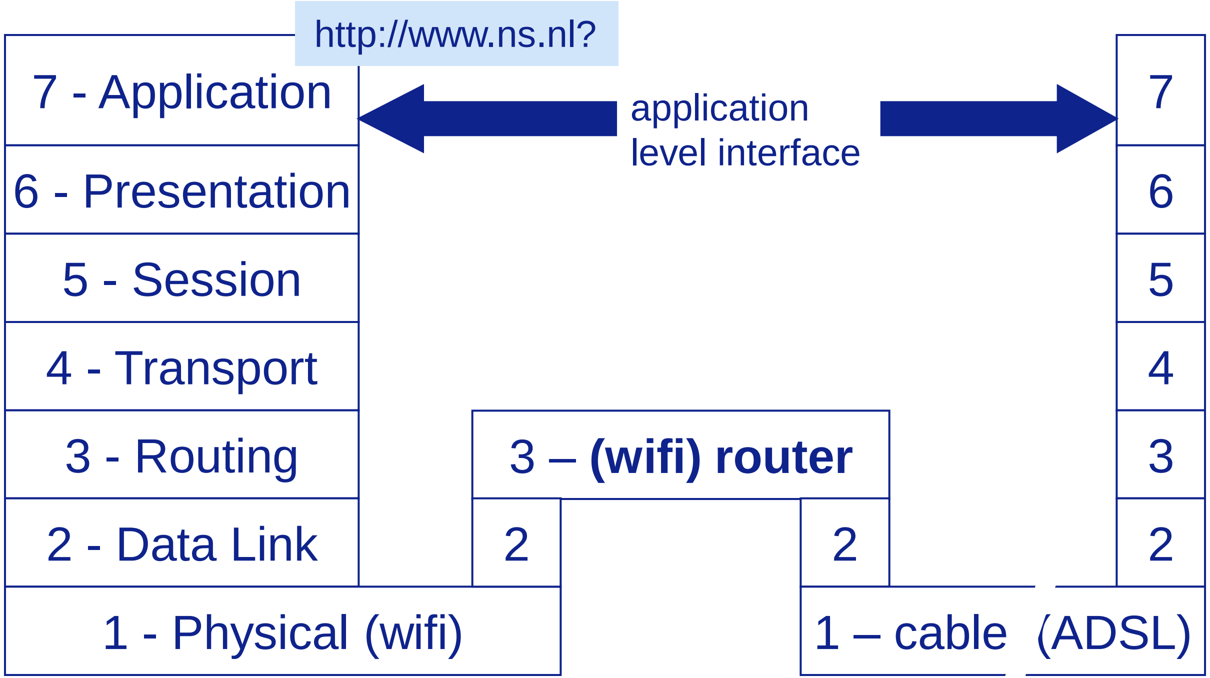


TCP/IP protocol suite	
Application Layer:	DNS · FTP · Gopher · HTTP(S) · IMAP · IRC · NNTP · POP3 · RTP · SIP · SMTP · SNMP · SSH · TLS/SSL · Telnet · UUCP · XMPP
Uses Session Layer:	ADSP · ASP · H.245 · ISO-SP · iSNS · L2F · L2TP · NetBIOS · PAP · PPTP · RPC · RTCP · SMPP · SCP · SOCKS · ZIP · SDP · SIP
Uses Transport Layer:	DCCP · SCTP · TCP · UDP
Uses Routing Layer:	ARP · ICMP · IGMP · IPv4 · IPv6 · RARP
Uses Data Link Layer:	ATM · Ethernet · FDDI · PPP · Token ring · Wifi · G3/G4

The ISO 7-layer communication model

Source: wikipedia

It is unusual to have a single physical link.
Example: wifi router, supposedly transparent….



Theory:

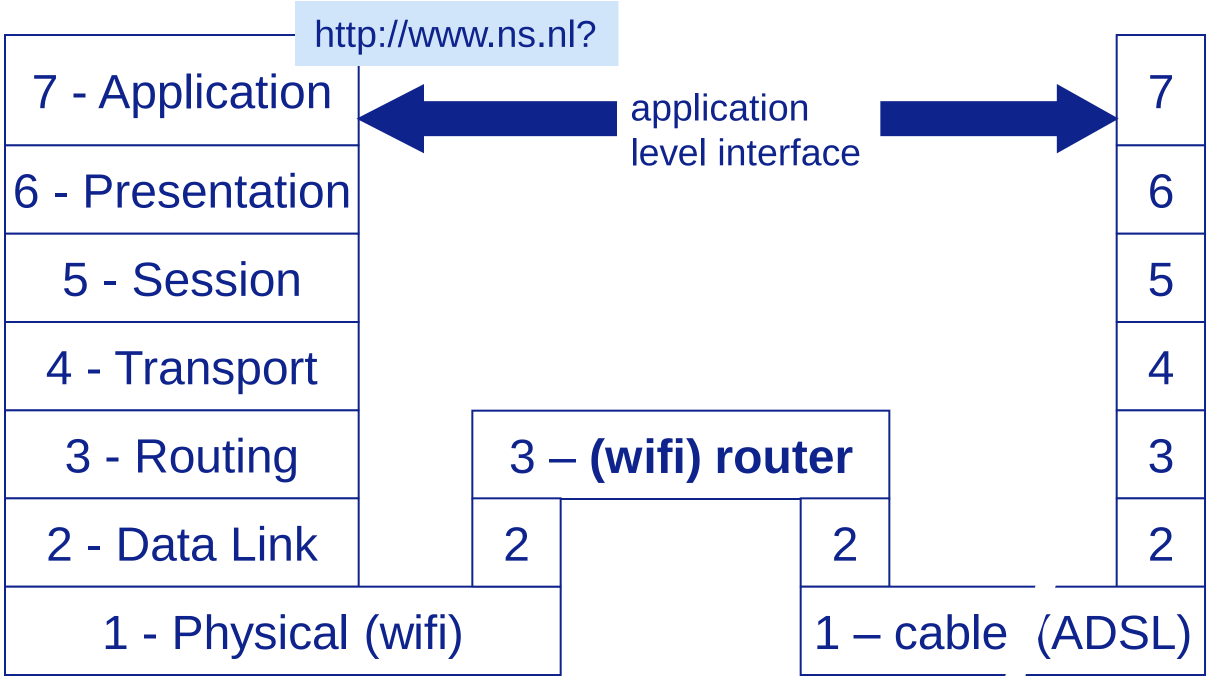
- Transparent

Practice:

- You know about wifi:
  - Connect
  - Login
  - Cost
  - Delay

(all ‘out-of-band’)

Routers can make you aware that protocols have multiple versions



Theory:

- Router is 100% transparent for http

Practice:

- Only http 1.1 or newer is supported
- Or 1.0 is just very slow….

Intermezzo: 112 emergency calls and G-versions

112 calls have priority on the GSM network.

But your phone does not know which network is available.

If it tries G4 first, with perhaps a poor connection,
then the emergency call could fail.

If it had tried G2 or G3 first, perhaps the call
would succeed….

Summary

Connectivity is about layering, protocols and topology

The SECS stack (1982/1995) sets an example for ‘machine’ connectivity

10 sets of functions should be part of any connectivity design

A reliable design requires a single point of control

So what is layer 8?

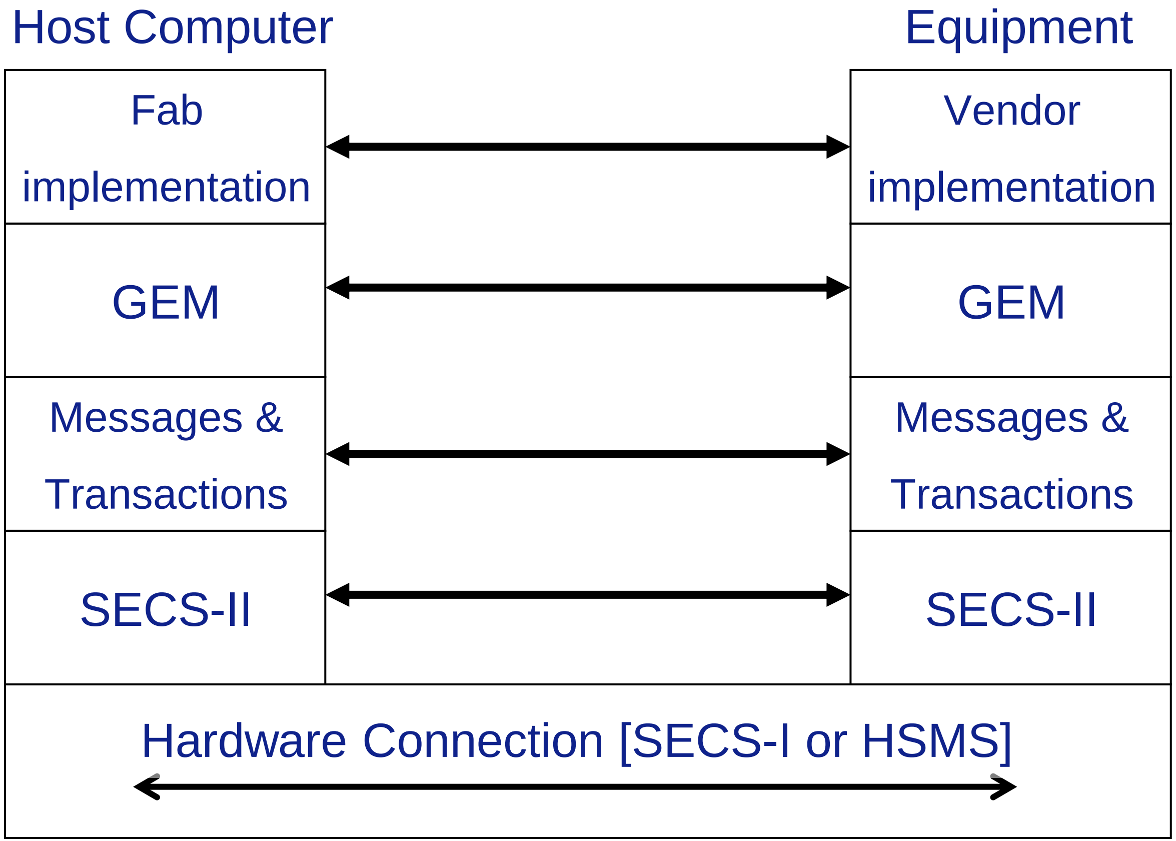
The SECS stack (1982/1995) sets an example for ‘machine’ connectivity

The SECS (Semiconductor Equipment Communication Standard)
standards are issued by the
SEMI (Semiconductor Equipment and Materials International) organization.

They survived the transition from RS-232 300 bits/s serial transport (1982) into the Gb/s TCP/IP era (from 1995 on).

While a standard for machine-to-machine communication (point-to-point), most concepts in it are relevant to any communication design.

The SEMI E* standards are organized per layer



Host Computer	Equipment
Fab implementation	Vendor implementation
GEM	GEM
Messages & Transactions	Messages & Transactions
SECS-II	SECS-II
Hardware Connection [SECS-I or HSMS]	

<- Eq. has subset of SECS standards.

E30:
Generic Equipment model:
must-haves + options

E5:
standard functions
and scenarios

Transport:

- E4: SECS-I: RS-232
- E37: HSMS: TCP/IP

SEMI E5 looks at ‘equipment’ from a ‘host’ perspective:

Stream 1	Equipment Status
Stream 2/17	Equipment Control and Diagnostic
Stream 3/4	Material Status and control
Stream 5	Exception Handling
Stream 6	Data collection
Streams 7/15/19	Recipe and parameter management
Stream 8	Boot Program Transfer
Stream 9	System Errors
Stream 10	Terminal services
Stream 11	Host File Services (deleted in 1989)
Stream 13	Data Set Transfer
Stream 14	Object Services
Stream 16	Running Jobs

Streams are like sections, and help organize the many functions in the standard

Now: you and your bank:

Section	SEMI E5 description	Your perception
1	Bank Status	Not available; assume functions are available when login is successful。 You get kicked out after 5 minutes idle without warning.
2/17	Bank Control and Diagnostic	Contact customer service (sometimes: chat available)
3/4	Material Status and control	Overview / order / configure your bank cards
5	Exception Handling	Various options (e-mail, SMS)
6	Data collection	Checking account: yes, Excel format, Credit card account: no
7/15/19	Recipe and parameter management	Periodic transfers
9	System Errors	Rare, but then handling is cryptic
10	Terminal services	Sometimes: chat available
13	Data Set Transfer	Download statements, tax overviews etc

Summary

Connectivity is about layering, protocols and topology

The SECS stack (1982/1995) sets an example for ‘machine’ connectivity

➡ 10 sets of functions should be part of any connectivity design

A reliable design requires a single point of control

So what is layer 8?

10 sets of functions to be considered for any connectivity design

Stream 1	Equipment Status
Stream 2/6/17	Equipment Control, data collection and Diagnostics
Stream 3/4	Material Status and control
Stream 5	Exception Handling
Streams 7/15/19	Recipe and parameter management
Stream 9	System Errors
Stream 10	Terminal services
Stream 13	Data Set Transfer
Stream 14	Object Services
Stream 16	Running Jobs

In SECS E5, host can always know equipment status and vv

Stream 1 Equipment Status

E5 is chatty.

- Host queries equipment status at intervals (heartbeat)
  - Reply reports a.o. current software release version
- Similarly equipment queries host at intervals

Spooling, if implemented and enabled, allows delivery of messages after a short interruption in communications.

- Host configures buffer size and can query spool buffer status

In E5 host computer is in control

Stream 2/6/17 Equipment Control, data collection and Diagnostics

If allowed by equipment operator, can switch equipment to remote control

- Host can then send commands (like ‘Run job’)
- Equipment guarantees reply in 30 seconds (will/will not run job)
- Actual job can take (much) longer

Host can configure reports to be sent at specific events:

- Host sends a list of variables (sensors, recipe name for job etc) and indicates for which events the report must be produced
  - This way host controls all data traffic in the network
- Equipment reports values, at intervals, or at certain events (e.g. job done)
  - Reports can be general, e.g. job progress, or detailed (diagnostics)

In E5 host can follow the material in the machine

Stream 3/4 Material Status and control

Host can see where the material is,
and can control the opening/closing of entrance ports etc.

This stream also allows the host to monitor and control the actions of robots that deliver and pick up material.

Finally the host can assign an identifier to
material that does not have a machine readable (bar)code.

In E5 the host can know about exceptions and alarms

Stream 5 Exception Handling

Host configures which exceptions and alarms must be reported
(and so controls the related data traffic)

When a selected alarm occurs, or is cleared, the host is notified.

Protocol, as standardized, is not qualified for safety-related alarms.

Recipe handling has evolved over time

Streams 7/15/19 Recipe and parameter management

Host starts a job by naming a main recipe and recipe parameters.

- Additional (sub)recipes can govern specific job tasks, e.g. material handling.
- Some equipment will allow recipe parameters to be changed while job is already running

In spite of all evolution, there is no standard for the recipe content.

- Most equipment uses a binary and proprietary format

Some (sub)recipes can be very large

- e.g. an image library of several Gb
- special functions are created to circumvent a maximum messages size fixed in the SECS-II protocol (16 Mb, considered a very large number at the time)

Error handling should be designed carefully

Stream 9 System Errors

Examples:

- failure from host or equipment to return an answer in 30 seconds
- Incorrectly formatted message received

A separate group of functions:

- By definition a system error is ‘out-of-band’
  - Indicating a failure in lower-level (or higher-level) protocol handling
  - In-band errors are signaled differently (e.g. OK/nok reply in a function)

Stream offers a set of standard replies (e.g. ‘Conversation timeout’) that can be sent if a normal reply is not possible, acting as a reset of the communication activity.

Nothing old-fashioned about text chatting

Stream 10 Terminal services (now called: chatting)

Originated in a time when telex was the only alternative
for the exchange of text messages.

In the SECS (fab) environment largely replaced

by other communication services (instant messaging, e-mail etc).

In other environments still good to consider:

- Instant chat between service person and machine operator
- Immediate contact with a customer service person
- Human communication in parallel to remote service (BRES)

An all-time favorite: copying files

Stream 13 Data Set Transfer

In itself nothing special – many protocols exist for file transfer like FTP

However, integrating ‘FTP’ in the SECS protocol
enables the host to fully control the transfer

- Can sync file transfer with e.g. starting a job
- System errors during file transfer reach the host via Stream 9

Object oriented programming across a network

Stream 14 Object Services

A late addition to the SECS protocol – added in the 1990's in parallel to the transition from 200mm to 300mm wafers.

Makes selected SW objects inside a machine visible to the host.

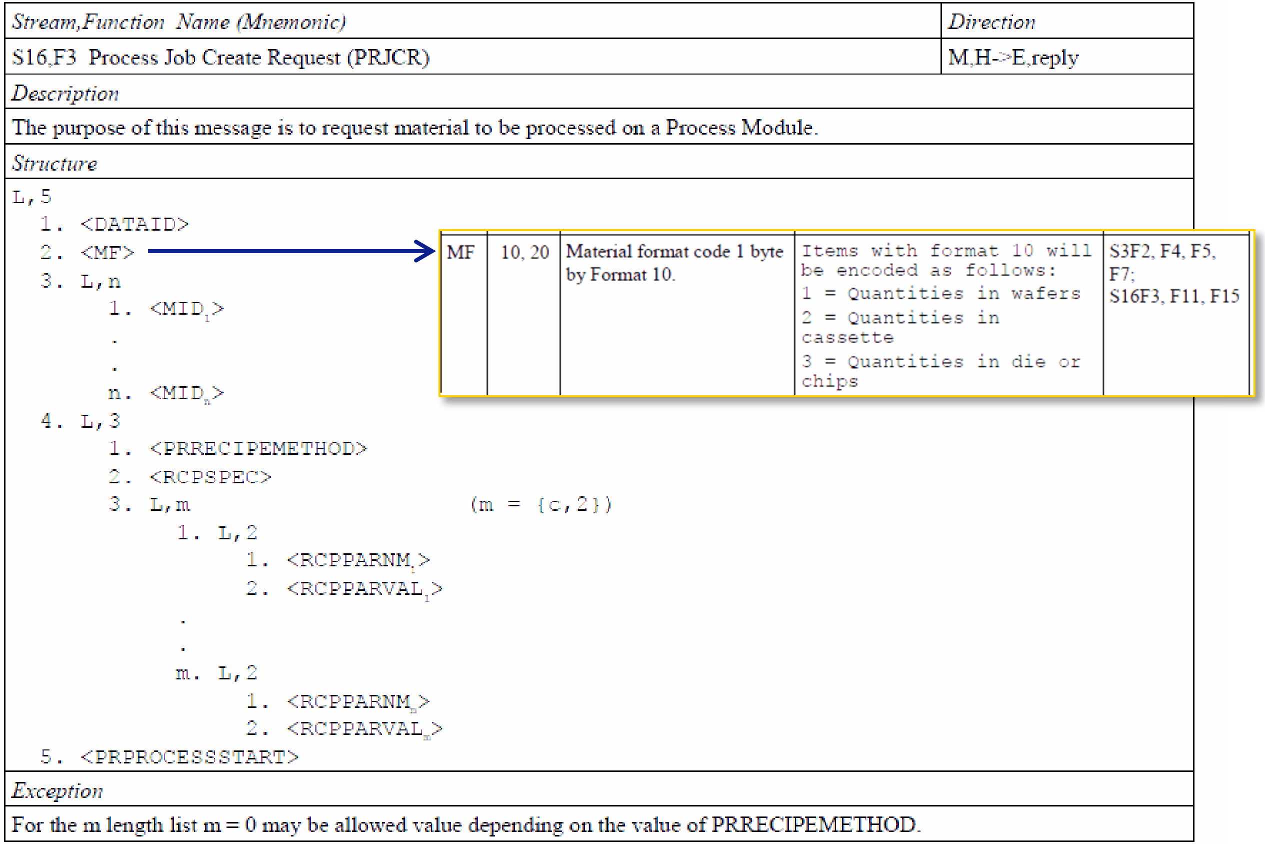
- Host can create or delete objects
  - Every object has a type (e.g. 'wafer') and a handle ('id')
- Host can read and/or write object fields (called 'parameters')
- Host can initiate actions on the object (e.g. 'run' a 'job' object)

Objects then became an enabler for additional automation standards
(e.g. E116, Equipment Performance Tracking: 'tracker' object)

Object oriented programming across a network

Stream 16 Running Jobs

Functions to create ‘Job’ Objects and start them running.
E.g. S16,F3



Stream,Function Name (Mnemonic)	*Direction*
S16,F3 Process Job Create Request (PRJCR)	M,H->E,reply

Description

The purpose of this message is to request material to be processed on a Process Module.

Structure

```
L,5
  1. <DATAID>
  2. <MF>
  3. L,n
       1. <MID1>
       .
       .
       n. <MIDn>
  4. L,3
       1. <PRRECIPEMETHOD>
       2. <RCPSPEC>
       3. L,m                       (m = {c,2})
            1. L,2
                 1. <RCPPARNM1>
                 2. <RCPPARVAL1>
            .
            .
            m. L,2
                 1. <RCPPARNMm>
                 2. <RCPPARVALm>
  5. <PRPROCESSSTART>
```

MF	10, 20	Material format code 1 byte by Format 10.	Items with format 10 will be encoded as follows: 1 = Quantities in wafers 2 = Quantities in cassette 3 = Quantities in die or chips	S3F2, F4, F5, F7; S16F3, F11, F15

Exception

For the m length list m = 0 may be allowed value depending on the value of PRRECIPEMETHOD.

Example: internet connectivity (so I should feel ‘in control’)

1	Website Status	tablet/laptop: some info, deeper: limited
2	Website Control, data collection and diagnostics	typically poor to average
3	Material Status and control	kind-of organized (e.g. amazon)
5	Exception Handling	no standard; HTTP does not have it
7	Recipe and management	unusual to see me creating one
9	System Errors	no standard; HTTP does not have it
10	Terminal services	sometimes - chatboxes
13	Data Set Transfer	ad-hoc standard (upload/download)
14	Object Services	no standard, but available (‘settings’)
16	Running Jobs	simple (1 provider): ok. but hotel+flight reservation: can fail miserably

Summary

Connectivity is about layering, protocols and topology

The SECS stack (1982/1995) sets an example for ‘machine’ connectivity

10 sets of functions should be part of any connectivity design

➡ A reliable design requires a single point of control

So what is layer 8?

A reliable design requires a single point of control

Example: reserve hotel + flight

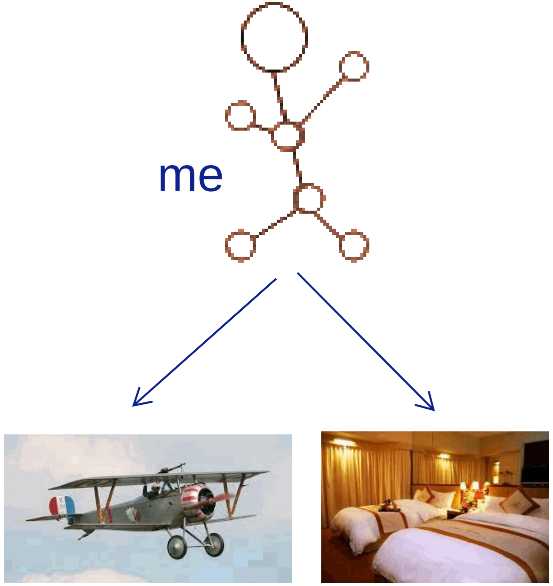
Two databases are involved: hotel + airline

- These are not linked, so I must orchestrate the 2 transactions
- But I cannot do an ‘atomic’ commit on 2 different databases
  - In fact, no SW system can today
- Consequence: If 2nd commit fails, I’m stuck with a room and no flight

If it was the last available room, and I cancel it, I’ll have no room and no flight…

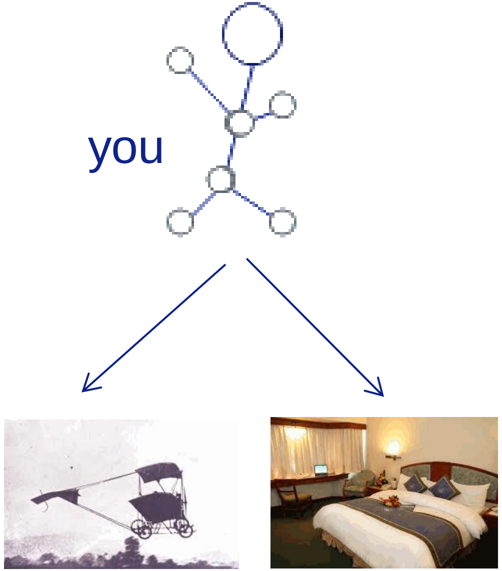
This does not feel like ‘in control’.

Intended outcome



me

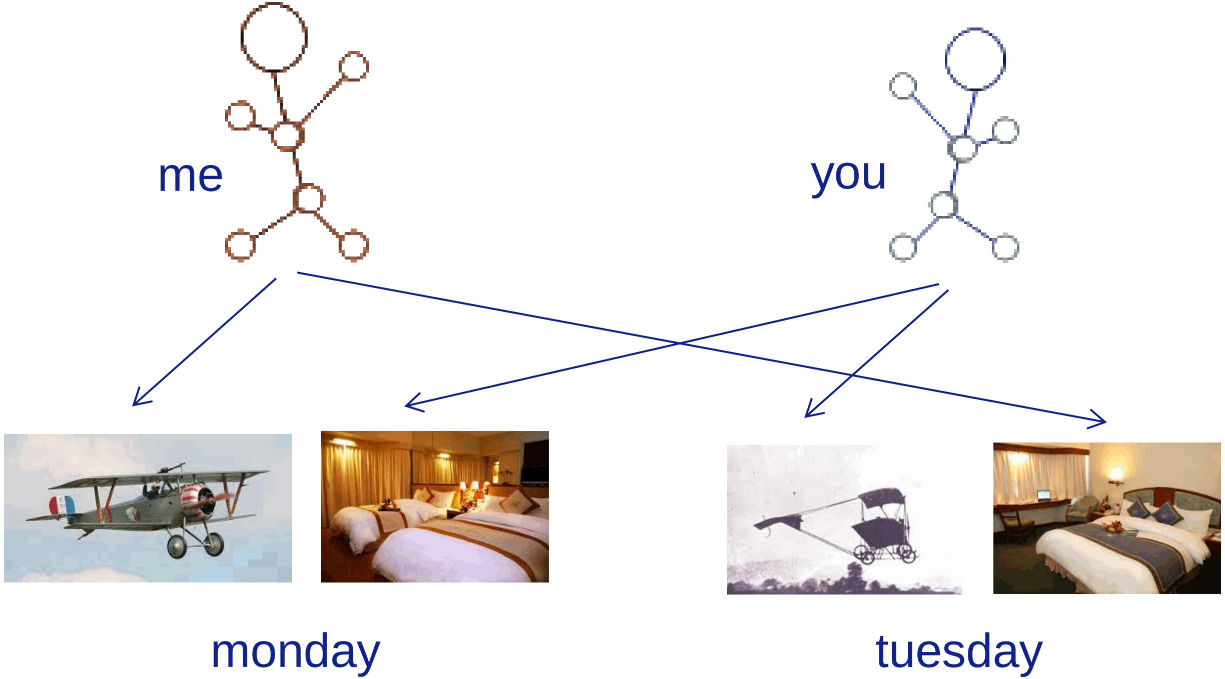
monday



you

tuesday

Possible outcome: Not OK for either of us



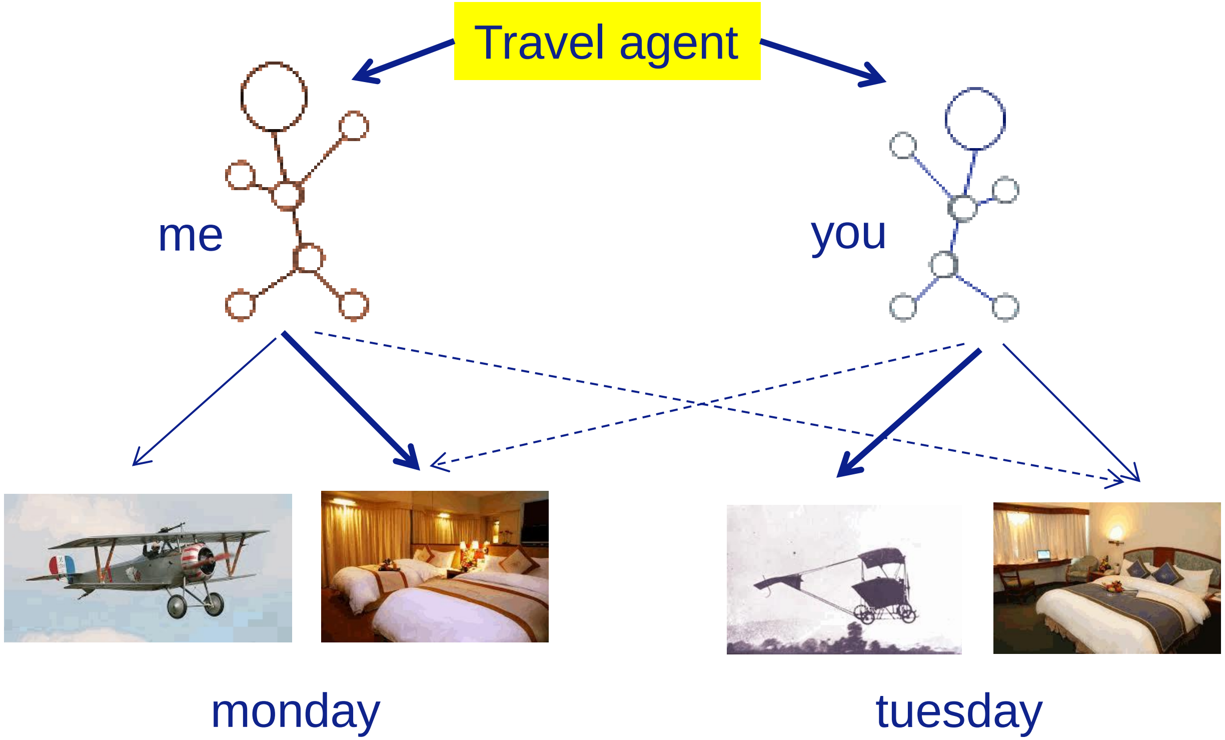
me

you

monday

tuesday

If there were a single point of control it/him/her could have brokered a swap:



Travel agent

me

you

monday

tuesday

Summary

Connectivity is about layering, protocols and topology

The SECS stack (1982/1995) sets an example for ‘machine’ connectivity

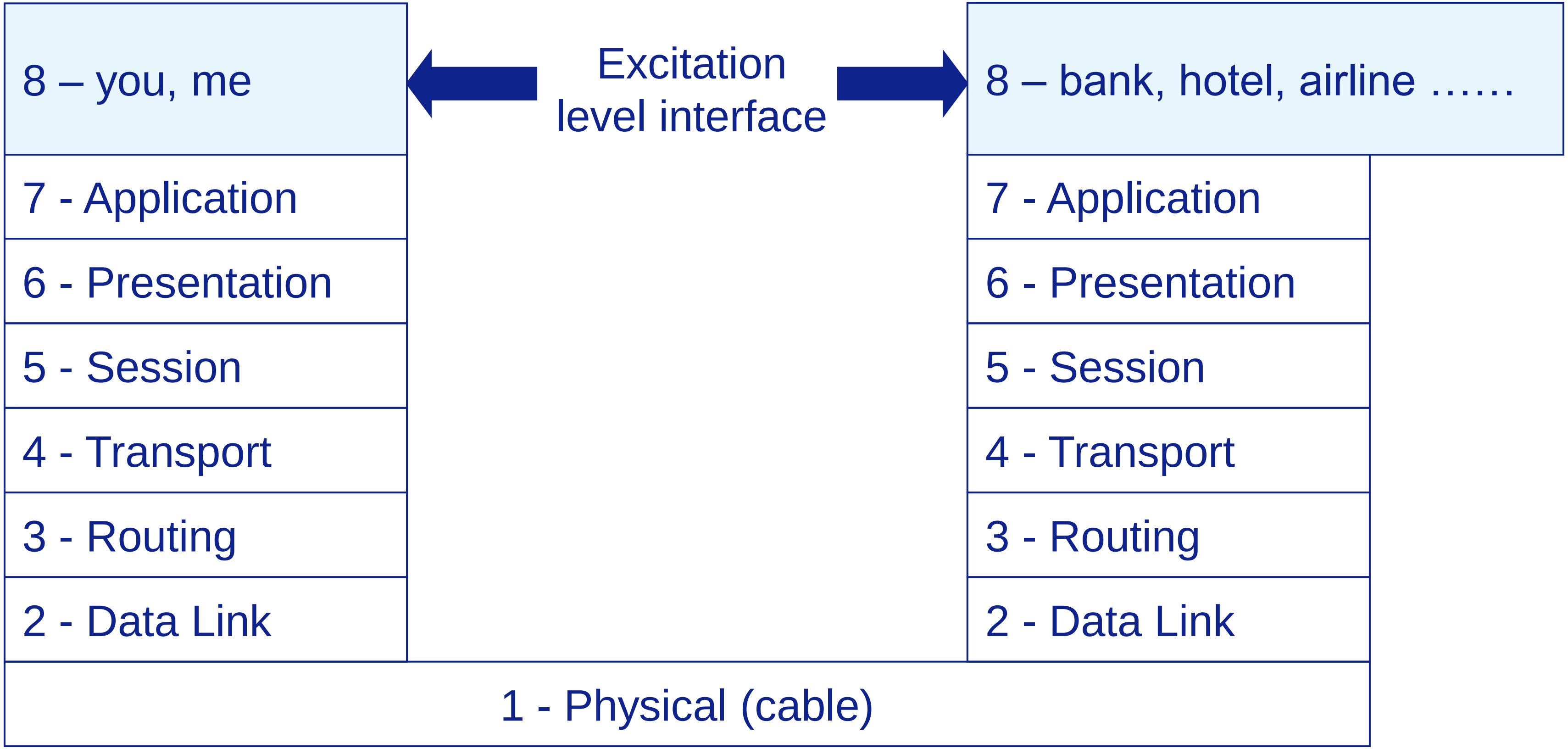
10 sets of functions should be part of any connectivity design

A reliable design requires a single point of control

➡ So what is layer 8?

So what is layer 8?

So what is layer 8?
Well… you and me connected to the rest of the world



8 – you, me

Excitation level interface

8 – bank, hotel, airline ……

7 - Application

6 - Presentation

5 - Session

4 - Transport

3 - Routing

2 - Data Link

7 - Application

6 - Presentation

5 - Session

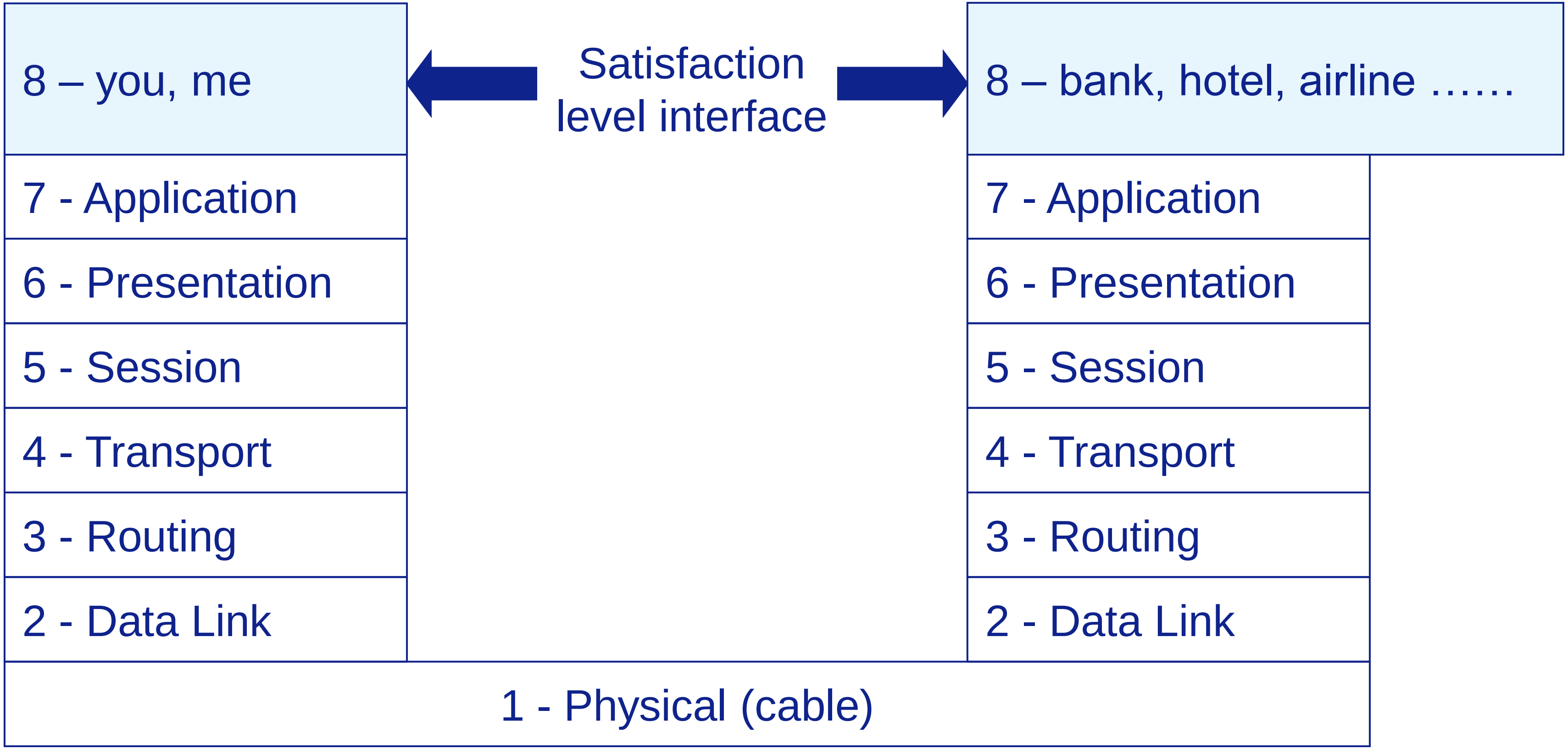
4 - Transport

3 - Routing

2 - Data Link

1 - Physical (cable)

Layer 8 is about my (your) level of satisfaction



8 – you, me

Satisfaction
level interface

8 – bank, hotel, airline ……

7 - Application

6 - Presentation

5 - Session

4 - Transport

3 - Routing

2 - Data Link

7 - Application

6 - Presentation

5 - Session

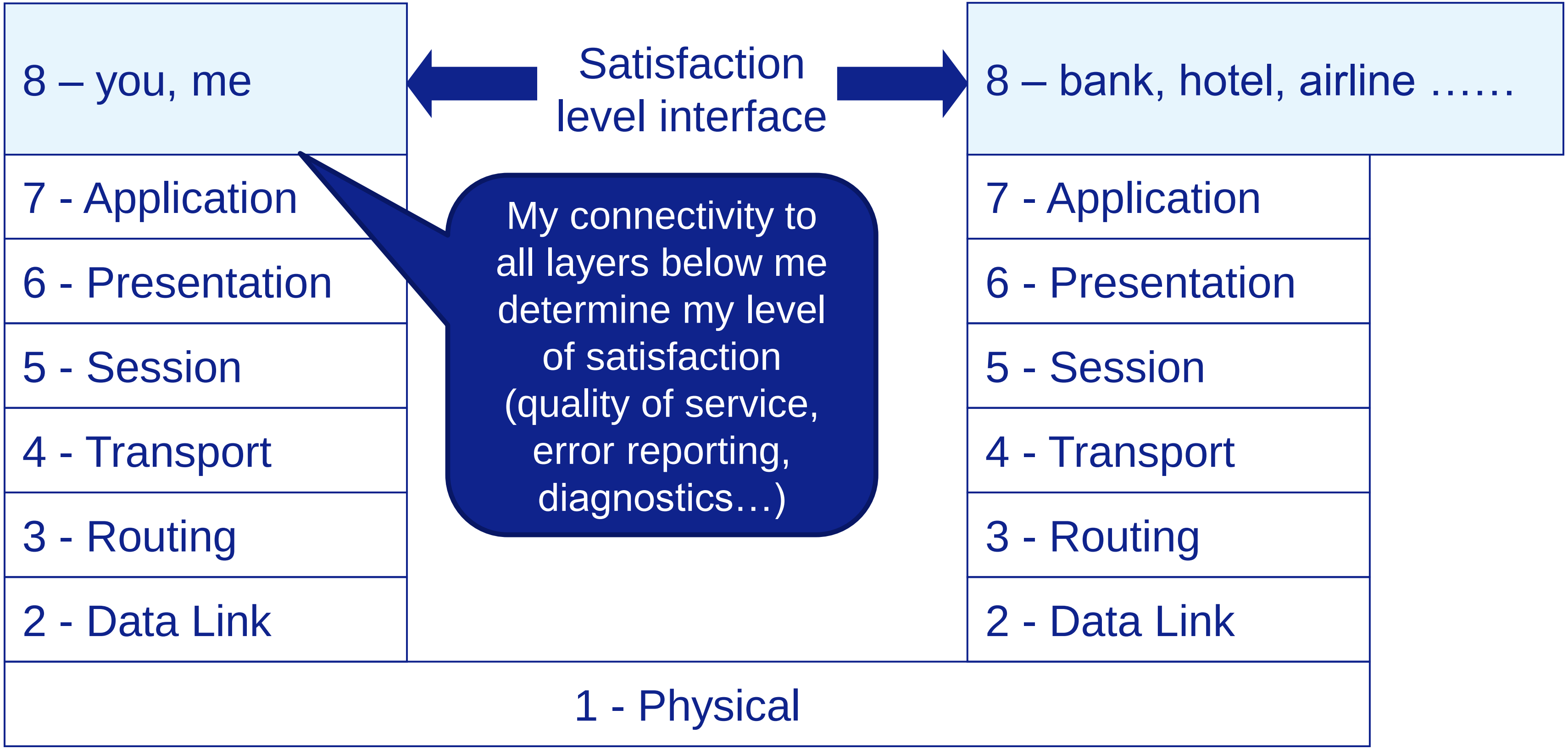
4 - Transport

3 - Routing

2 - Data Link

1 - Physical (cable)

Layer 8 is about my (your) level of satisfaction



8 – you, me

Satisfaction level interface

8 – bank, hotel, airline ……

7 - Application

6 - Presentation

5 - Session

4 - Transport

3 - Routing

2 - Data Link

7 - Application

6 - Presentation

5 - Session

4 - Transport

3 - Routing

2 - Data Link

1 - Physical

My connectivity to all layers below me determine my level of satisfaction (quality of service, error reporting, diagnostics…)

Final Summary

Connectivity is about layering, protocols and topology

The SECS stack (1982/1995) sets an example for ‘machine’ connectivity

10 sets of functions should be part of any connectivity design

A reliable design requires a single point of control

So what is layer 8?

Conclusions and discussion topics

Can you / do you want to subscribe to this statement?

“ As a matter of workmanship,
I do not accept a connectivity solution unless:

- It has a defined topology (point to point, star, multi-connected)
- It has a clear layering
- Protocol(s) are selected per layer
- Provisions are made to allow for newer protocol versions to be used in a later stage
- Authentication and authorization is at the right layer (or layers)
- The end user (person/computer) is in control”