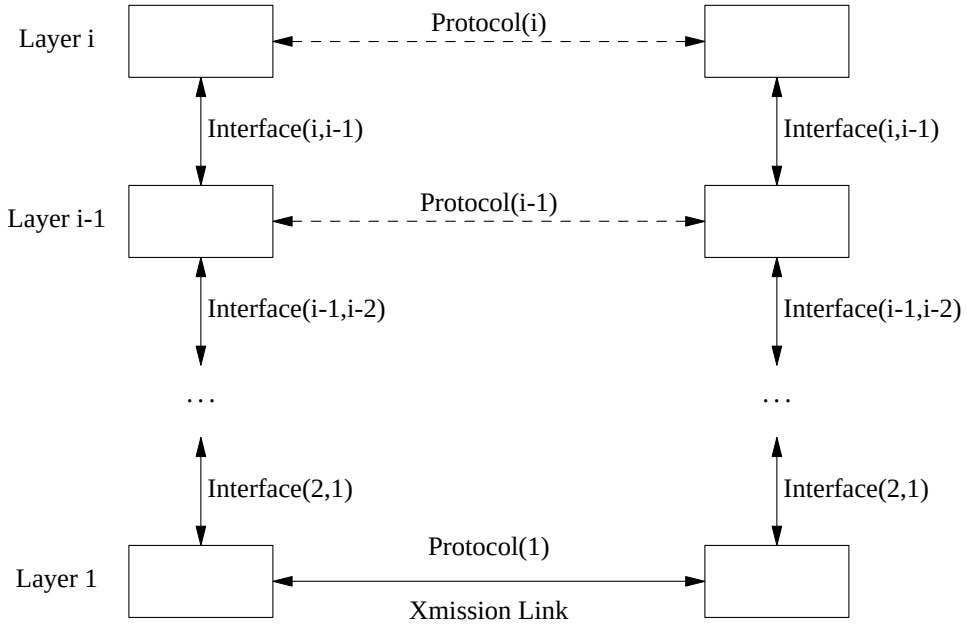
Chapter 2: Layered Architectures

1. What is a Layered Architecture?

Layered architectures divide the work that communications protocols must perform into distinct, separable components.



Components of layered architectures

1. The solid lines indicate paths that bits flow over.
   The dashed lines indicate two modules that communicate, using the solid path.
2. Protocol(i)

   Defines the rules that two modules in the same layer use to exchange data

   Example

   - Set-up a connection
   - Xmit data
   - acknowledge data
   - retransmit data
3. Services
   What functions layer i performs for layer i+1

   A. Example; Channel Sharing

   Channel sharing protocols may
   1) perform TDM, FDM or CDMA to multiplex several users onto a single channel,
   2) break messages into packets and place them in a queue for a statistically multiplexed network, or
   3) implement CSMA/CD to share the channel in a LAN.

The service that these protocols provide is a communications channel, and the protocols above this layer are not concerned with the intricacies of acquiring a channel.

Internet services operate over all types of networks because they operate above the layer that provides this service.

B. Example: Reliable message delivery.

If there is a protocol at layer i that recovers missing messages and places those messages in the same order, then all of the protocols at layer i+1 and above can assume that they operate over a reliable channel with no transmission losses.

The protocols at the higher layers are simpler than they would be if they also had to implement an error recovery procedure.

4. Interfaces

The rules for transferring data and commands between layers i and i-1

— What the data means

— How the services are selected
1) There may be services provided by a layer that are not needed for a particular application
2) There may be several different protocols at a layer, for instance in example A above, there may be several different channel sharing protocols, but only one is selected depending on the network.

— Information when the service cannot be provided

5. Data Encapsulation: Data Hiding

Layer (i) Header	Packet from Layer (i+1)	Layer (i) Trailer

— The header and trailer are added by a layer to implement its protocol and passed to the lower layer

— It encapsulates everything from the higher layer

— Data from the higher layer is unchanged

— The header and trailer are stripped off before passing the data to a higher layer

— The operation of the lower layer is hidden from the upper layer

2. Why a layered architecture?

1. Simplify the protocol implementation
by breaking the protocol functions into smaller pieces, each of which performs a well defined function.

- In a layered architecture a programmer who writes an application passes the data to a program that is responsible for passing the data over the network, rather than controlling the network directly.
- Layered architectures hide the network functions from the application.

- The application programmer must only be concerned with the messages that are passed between the distributed application, rather than also being an expert on networks.

2. The network can change without changing the application programs.
   - This has made it possible for the Internet to change from a microwave network with dedicated 50 kbps links, to an ATM network, back to a packet network, and eventually to a DWDM network, with multiple 10 Gbps channels on each fiber, without the application programs changing.
   - The Ethernet has changed from a CSMA/CD network to a switched network.
3. Re-use software to decrease the time needed to implement new services
   - Many network applications require protocols that perform the same functions.
     — For instance, many applications must guarantee that the data is correctly received.
   - Rather than engineering a new protocol for each application, writing the code to implement the protocol, and testing the code, standard protocols have evolved and the software to implement these protocols is reused in every application.
     — In the 1960's network applications placed the data on the communication channel and were responsible for implementing all of the protocols that were needed for error control, selecting a channel or route, implementing flow control, ...
     — A programmer who wrote a program to transfer funds between two accounts in different banks had to know about all of the network functions that were needed to operate the network that would be used.
4. Protocol verification and conformance testing is simplified.
   - For verification we assume that the lower layers are correct and only test that a protocol at layer i provides its service if the protocols at the lower layers correctly provide the services they advertise.
   - For conformance testing we test the sequences of inputs that can come from the layers above and below this layer, rather than testing all of the possible input sequences in the system.
     In most cases, the number of output sequences from a protocol is much less than the number of input sequences.
     For instance, in the error recovery protocols there are many combinations of lost and recovered messages that lead to the same sequence of delivered messages.

2.1 History of Network architectures

1. The early protocol architectures were company proprietary and unified the company's products, so that different products from the same company could be mixed together.
   Examples:
   DecNet - DEC Computers
   SNA - Simple Network Architecture - IBM
2. The ARPAnet developed the TCP/IP architecture, that was used throughout the network, and allowed many independent researchers to contribute to the network development.
3. Standards Organizations
   ISO/OSI - International Standards Organization/ Open System Integration
   — Objective: To unify the different architectures to allow interoperability
   — The military tried to make this the only standard architecture in the 1990's by only purchasing equipment that used the architecture, but failed because of the large commercial market for Internet equipment.

— This architecture is still used to discuss different protocols.

2.2 How should we select the layers?

1. A layer for each hardware component that the message passes through
   — This is a natural separation - each hardware component provides a service
   — It allows us to upgrade new hardware
   ie. to change from telephone line modems to fiber optic modems without changing the rest of the network programs.
   — A typical hardware layering is:
      — A data link layer - to control the modems
      — A network layer - to operate the network switches and multiplexers
      — A user layer - to operate in the end user devices
   — Hardware layering is the basis for the TCP/IP protocol on the ARPAnet
      — IP is implemented on IMP's (todays routers)
      — TCP implemented on hosts
2. Layers for different functions that are performed on the same piece of hardware.
   — This became more common with the arrival of complex, inexpensive general purpose computers that replaced several pieces of hardware
   — It allows some functions to be changed without changing all of the protocols
   — It is the basis for the ISO/OSI architecture
   — In today's Internet, some end user devices also perform routing.
3. Functional Architectures
   — Functional architectures are an alternative to simple layered architectures.
   — They have separate modules that perform specific protocols for error control, flow control, admission control, routing, ...
   — Instead of having a linear connection of protocols, that must be performed in a specific order, they can use the protocols in different orders, and possibly several times.
   — These architectures are currently being discussed for high performance networks that have high transmission rates or networks with very different error models.
   — These architectures are justified because the same protocols are sometimes used at different layers of the current architectures.
   — For instance,
   Which layer should perform retransmissions of packets that are in error?

   end-to-end: The Internet assumes low error rates, typical of fiber-optic links, and performs error control at the transport layer

   hop-by-hop: Packet radio networks have higher error rates than fibber-optic networks and typically performs error control at the data link layer
   Packet radio networks also use smaller packets than fiber optic networks to increase the probability of successful transmission

   In general, the same retransmission strategy may be called by different layers in the same network
   We can consider changing the layer that performs retransmissions as the error rates in a

network change.

3. The OSI reference model

— Used to discuss, think about, and design protocols
- very seldom used as an implementation
- it was tried, but it is usually too complex

— Peer protocols are protocols in the same layer

— they communicate by exchanging PDU's - protocol data units

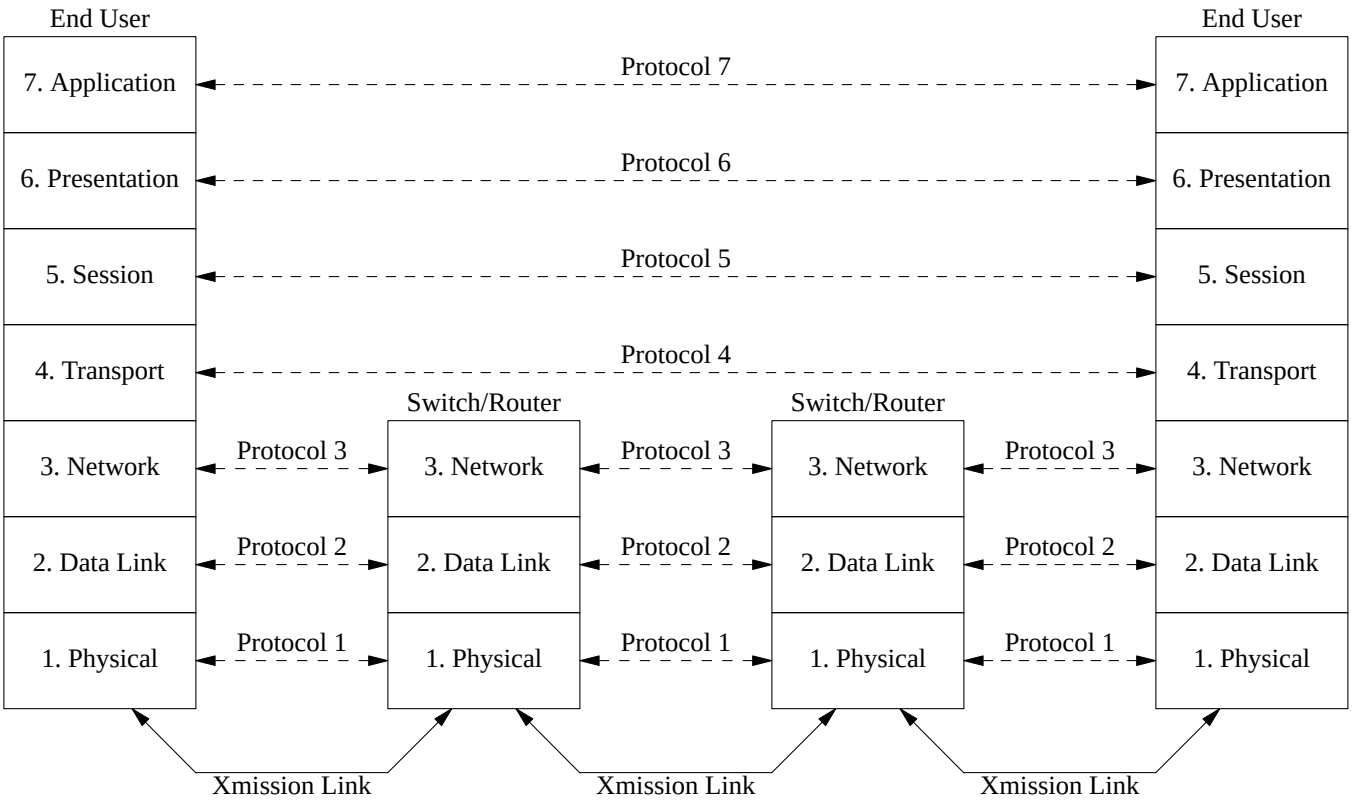
— A PDU consists of a header to control the the operation of the protocol and an SDU - the service data unit

— protocols in the upper layers don't communicate directly, but use the services provided by a lower layer protocols

— the services are negotiated across an SAP - service access point
- the SAP may set up a connection for a connection oriented service or accept the data directly

— once the service is specified, a layer N protocol accepts a PDU from the layer N+1 protocol and encapsulates it in its own PDU as the SDU
the PDU from layer N+1 becomes the SDU at layer N

3.1 The 7 layer architecture



— 7 - application layer

- the services that are provided to the user - http for the WEB, SMTP, FTP, remote login, network management, ...

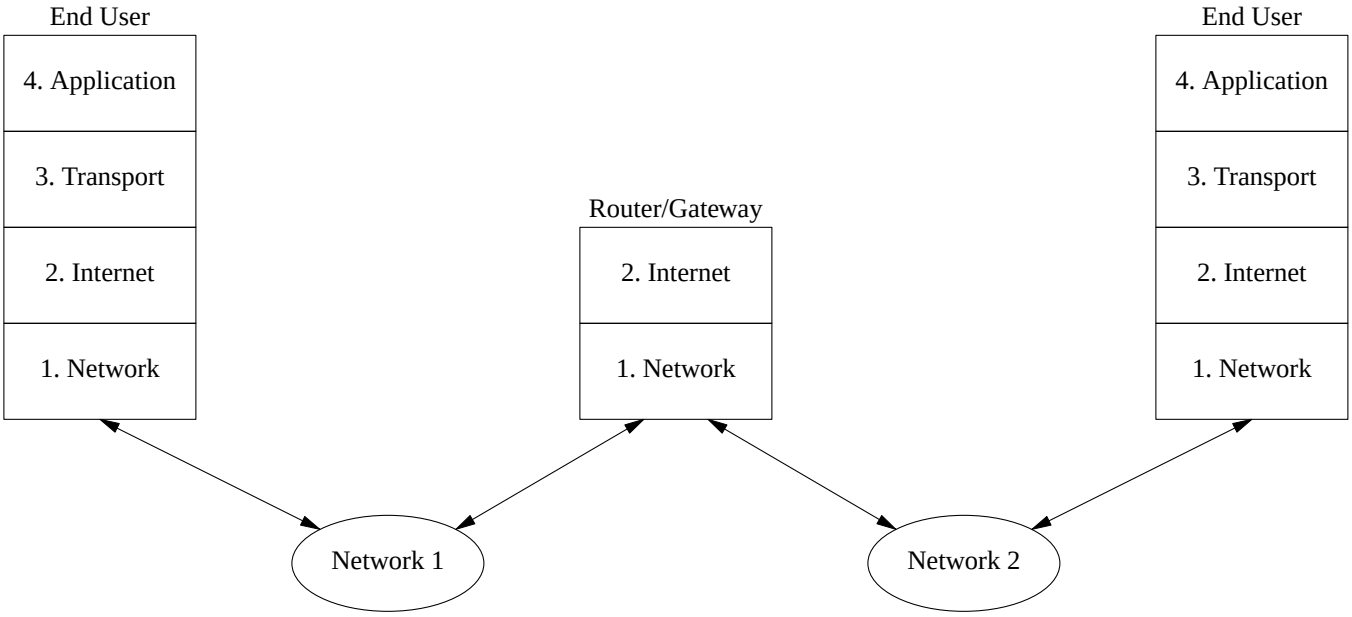
- 6 - presentation layer
  - Convert the different representations used by different machines into a common format
  - more important in the early days of computers when there were different byte and word sizes in different computers
    IBM: 8 bit bytes, Univac 9 bit bytes (36 bit words), DEC 6 bit bytes
  - Xlate different data formats to exchange funds between banks
  - ASCII -> EBCIDIC
  - At present this layer is used to translate between security formats.
- 5 - Session Layer
  - controls the transmission of information for half duplex or full duplex communications
  - check-pointing to transmit long blocks of data over an unreliable medium
- 4 - Transport layer
  - responsible for the end-to-end transfer of data
  - sets up a connection for connection oriented transfers
  - break long messages into packets or cells that can be transferred by the network layer - packet segmentation and reassembly
  - multiplexing several applications for efficient transmission - assembling packets for packet voice.
- 3 - Network Layer
  - Transfers the data from node to node to get a path through the network
  - Hides the specifics of the network from the upper layers
  - Inter-networking if we have to cross networks that use different technologies ie. Internet, telephone network, an ATM network and local networks
    convert to the formats expected by each sub-net that the data traverses
  - Congestion control
  - Routing
- 2 - Data link Layer
  - transfer the data across one link in a network
  - HDLC, PPP,
  - the access protocols used in LAN's - MAC protocols - media access control protocols
- MAC layer: combination of layers 1+2
  - There may still be a data link layer above the MAC layer
- 1 - Physical Layer
  - transmission of bits over the physical links
  - wire line, CATV, radio, ...

4. TCP/IP architecture

reference [1] section 2.3

- The dominant architecture today
- More closely tied to a specific network - the Internet
- Bottom up development of the architecture - instead of starting with a model then specifying protocols it started with the protocols that were needed on a specific network and then organized them into layers depending on the hardware that used the protocols

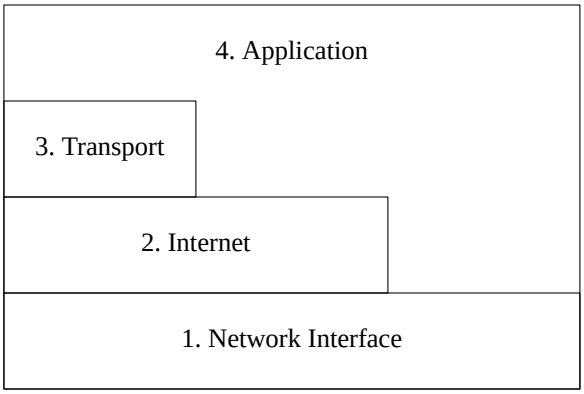
4 layer architecture - reference [1] figure 2.11 pg 58



- 4 - Application layer:
  - layers 5,6,7 in the ISO model
  - http, smtp ...
- 3 - transport layer
  - TCP - transmission control protocol - error control, connection, resequencing, congestion control
  - UDP - user datagram protocol - best effort - no guarantees - no retransmission
- 2 - Internet Layer
  - the IP protocol
  - routing across the network
  - datagram - best effort delivery of packets
  - also congestion control
  - layer 2 and 3 cover layers 3 and 4 in the ISO architecture

- 1 - Network layer
  - transfer the data over over a specific network
  - Packet network links, can also be a switched network like the telephone network or an ATM network
  - added when the Internet evolved from the DARPA/NSF net
  - the objective was to allow the Internet to survive changes in the transmission network and also to use new transmission technologies in parts of the network without changing the entire network
  - The Internet became too expensive to change for each new technology

Not a true layered architecture
reference [1] figure 2.10

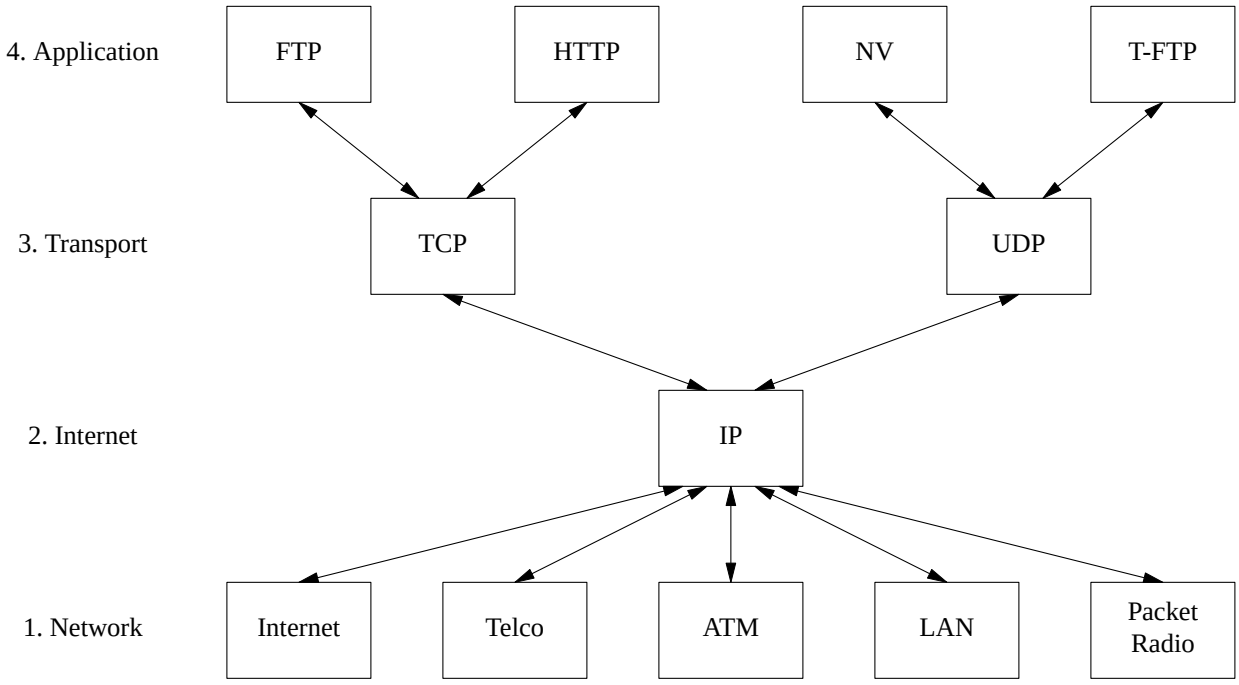


- Applications can bypass the layers
- Advantages
  - More efficient - bypass the layers that aren't needed rather than trying to get around them
  - Get services that are available on lower layers, but aren't specified through the upper layers - ie. the QoS in an ATM network isn't specified in the transport layer or the IP layer
- Disadvantage
  - An application that skips layers may have to be rewritten in a different environment
  - An application that directly accesses PPP or an ATM network won't run over the general Internet

4.1 TCP/IP Hour glass

— reference [2]

— Nothing above IP worries about transmission technology, and nothing below IP worries about services - as long as we don’t bypass IP.



5. Architectures for Cyber-Physical Systems"

5.1 Reasons for Complexity

We found that there are several reasons why the verification and testing of cyber-physical systems is more difficult than solving the same problems for communications protocols.

1. Cyber-physical systems interact with the physical world in many ways.

   Whereas communications protocols only interact with the physical world through the communications channel. Each Interaction with the physical world has its own failure modes, and may detect infrequent events occurring in the physical world. For instance, a communications channel may lose messages, while a sensor may generate inaccurate measurements or may detect an obstacle on a roadway.

2. Most cyber-physical systems, particularly those used in automotive applications, are time critical.

   FSM's handle timing badly. We have verified communications protocols,modeled as a FSM, with a single timer for retransmissions, but we have not verified protocols with multiple timers. Other techniques, such as timed automata, have been developed to address the timing problems, but most of these techniques operate by determining all sequences of the timed events, and are limited in the number of timers or the number of participants that they can consider.

3. The number of participants in a cyber-physical system is larger than in many communications applications.

   Most communications protocols that have been verified are two-party protocols. In automotive applications, some of the protocols that we have verified have 8 or more participants. The potential size of the composite FSM state space, that must be verified grows exponentially with the number of participants.

5.2 Multi-Dimensional CPS Architectures

The architectures that we are investigating are multi-dimensional, with one dimension for each interaction with the physical world [3]. The dimensions are organized into stacks, similar to the stack that is used in communications networks, with the highest level of the stack, the cyber application at the origin, and the individual interfaces to the physical world furthest from the origin. An example of a 4-dimensional architecture, applied to cooperative driving systems, is shown in the figure. The dimensions may be loosely or tightly coupled, depending on whether or not the stack in one dimension uses the services from another, and how far from the origin the connection occurs. The further from the origin the connection, the more tightly coupled the physical processes.

For instance, the sensor stack in the figure may use the group communication protocol in the communications stack to create a common view of the sensors in each vehicle. This interaction is close to the origin, so sensors and communications are coupled, but not tightly coupled. The MAC layer protocol can change without changing the sensor coordination protocol. The IEEE 1588 protocol in the timing stack may directly access the IEEE 802.11 protocol in the communications stack to synchronize the timing in two of the vehicles. The interaction is further from the origin than the interaction between the sensor and communications stack, so that the timing and communications stack are more tightly coupled. There are fewer pieces of the communications stack that can be modified without modifying the timing stack.

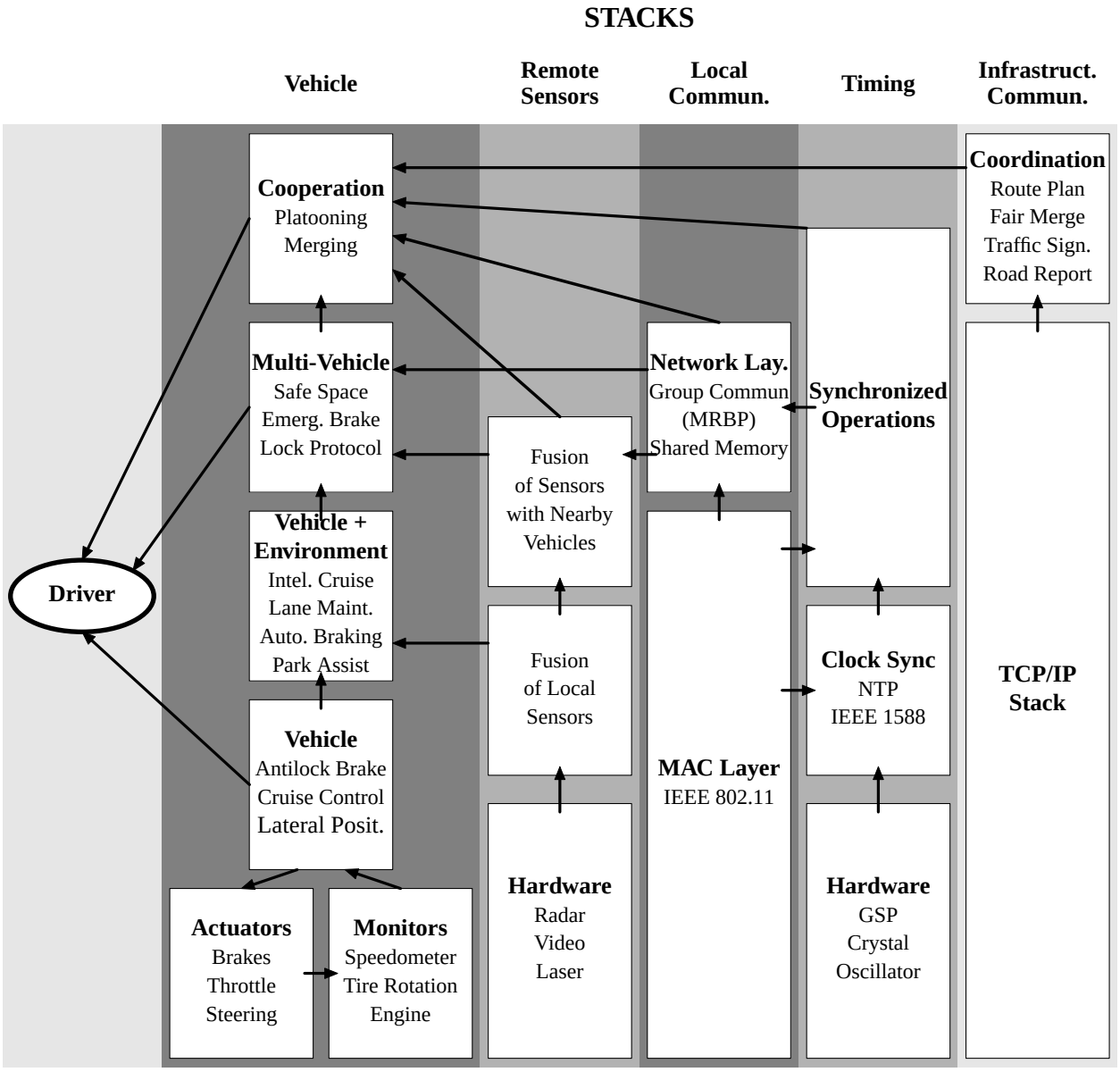


Figure 1. An architecture for intelligent vehicles

The incentive to create stacks in this architecture is similar to communications networks, they make it possible to separate the higher level logic functions from the physical implementations. By having well defined interfaces between the layers in the stacks we can change a layer without changing the entire stack. For instance, we can change the type of sensor that we use to detect nearby vehicles without changing an automatic cruise control protocol that uses the distances, as long as the sensor stack provides the same service, the same positioning guarantees. If the new sensors provide the old service, but also improve positioning we can use the new service to improve the applications, perhaps by having vehicles travel closer together.

In [4] we simplified a collaborative driving protocol by removing functions that appeared in multiple locations from the protocol, ans assumed that they are provided by a lower layer in the stack. This is similar to using subroutines in a program. When functions, such as providing specialized communication guarantees, are required in multiple locations within an application, or in several applications, the function is removed as a subroutine. The subroutine is a protocol that provides a service to the application, and is

pushed into a lower layer of the stack. The use of subroutines makes it easier to write an application, and when we find a better way to provide the subroutine's service, the improvement is realized everywhere that the subroutine is called.

In order to realize the improvements that are possible in writing applications, it is important that the definitions of the layers remain flexible, so that we can define new subroutines. This is in conflict with the objective to create well defined interfaces between the layers to minimize the effect of introducing new technology or algorithms. Similar conflicts occurred when the 7-layer ISO/OSI architecture was applied to the Internet. Too much structure inhibited change, and the simpler Internet architecture prevailed. However, it was necessary to introduce the Internet Layer into the Internet architecture so that new transmission technologies, such as ATM, optical switching and wireless local distribution, could be introduced without affecting Internet services.

5.3 Absolute Time

A major difference between protocols that are written today, and those that were written as little as 5 or 10 years ago, is the availability of inexpensive, accurate clocks, [5]. This is particularly true for automotive systems, where GPS devices have become common place, and provide clocks that are accurate within 10's of microseconds [27 – 32], and accurate crystal oscillators can maintain a clock when a GPS signal isn't available. The timing stack can be based upon GPS and a crystal oscillator to maintain time when a GPS signal isn't available, the network time protocol [12, 13], or the IEEE 1588-2008 standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.

Incorporating time into protocols has been investigated extensively [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]. All of the approaches use timers, rather than absolute time. Most determine the possible sequences that result from timers being set differently in different component machines.

Different sequences of timers are particularly troublesome when timers are set by a message that is transmitted over unreliable communications channels. If two participants in a protocol set identical timers when they receive a message from a third participant, and the first recipient receives the message on the first transmission attempt but the second participant requires two transmission attempts, the first participant's timer will fire first. However, if the second participant receives the message on the first attempt, and the first participant receives the message on the second attempt, the the second participant's timer will fire first.

Our objective is to simplify timed protocols by allowing operations to occur simultaneously, and to reduce the number of sequences that we must consider when verifying the protocol. In the next section we demonstrate how synchronization can simplify protocols by presenting a simple lock protocol that would be very difficult, if not impossible, to construct by setting timers over an unreliable communications channel. In the verification chapter we will explain how we can increase the safety of CPS by decreasing the number of possible execution sequences.

Simply using absolute time, rather than timers isn't sufficient to eliminate the number of execution sequences. If the participants participants set deadlines depending upon when they reach certain bench marks, they may set the deadlines differently. For instance, in the merge protocol vehicles may schedule the merge process to terminate, whether or not the driver has elected to merge, 10 seconds after the gap between cars is reached, If different cars detect a safe gap at different times, they may set their deadlines differently.

We are investigating several techniques to avoid multiple sequences. The first is to allow only one participant, the master, to set deadlines. The second, which fits well with our model of common data sets, is to maintain a common time matrix that every participant uses to set its deadlines. We are currently investigating rules for setting and distributing the matrix. The time matrix may significantly simplify conformance testing, as we will described in conformance testing chapter.

5.4 A Simple Lock Protocol

The merge protocol operates by creating a communication group consisting of a car that wants to merge and the two cars that it wants to merge between. The protocol is initiated by a driver signaling his intent to change lanes. The two cars in the adjacent lanes use their intelligent cruise control to create a safe gap for the merge, while the initiating car uses its intelligent cruise control to position itself in the gap. An intelligent cruise control system allows a driver to select a maximum cruising speed, but reduces that speed in order to maintain a safe distance to the car in front. Furthermore, the safe distance is adjusted based upon the speed and road conditions. Larger distances are maintained when the road is wet or icy. We assume that the intelligent cruise control can also be instructed to adjust the gap so that it is large enough to accept the merging car, and that the merging car can set its gap to the vehicle in the adjacent lane, as well as the vehicle directly in front.

If the two vehicles cannot create the gap, because of other vehicles or the road conditions, or the merging vehicle cannot position itself, possibly because of the movement of the vehicles in its own lane, the driver is given a red signal. The driver will also receive a red signals if any of a number of failure conditions occur. If a driver receives a red light in response to his turn signal, he must retry his merge at a later time. He only receive a green light when it is safe for him to change lanes.

At the beginning of the merge process, the merging vehicle obtains locks from the two vehicles in the adjacent lanes that will participate in the merge. The lock guarantees that the vehicles are cooperating with the merging vehicle, and are not in the process of initiating their own merge or are cooperating with any other vehicles. For example, in the figure car 1 wants to merge between cars 2 and 3. Cars 2 and 3 may also cooperate with cars A, B, C, D, and E, or may initiate their own merge.

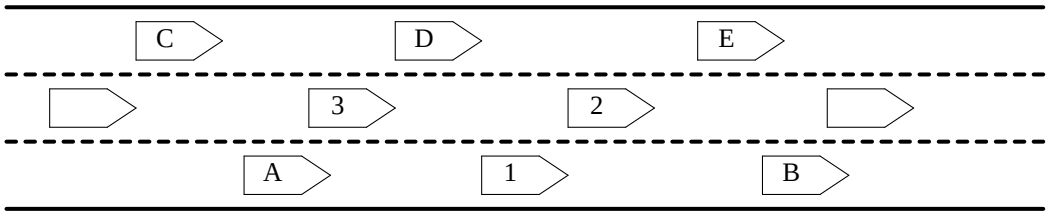


Figure 2. Merging Vehicles on a Highway

The objectives of the lock protocol are to;

1. guarantee that the three cars engaged in a merge protocol are only participating in one merge process at a time,
2. guarantee that, once the three cars are locked, none of the three releases the lock before any of the others, and
3. guarantee that the three cars eventually release the lock, even if the communication channel fails or one of the locked cars leaves the roadway, so that all of the cars can participate in future merge processes.

The figure below is the FSM of a simple protocol that meets the objectives. The car that initiates the merge sets a common deadline in all of the vehicles. The lock is only considered granted, and the protocol progresses, if the initiator receives the grant from the other participants. Whether or not the protocol progresses, all of the participants that have granted the lock release their locks simultaneously. The protocol meets the requirements independent of which messages are lost. Achieving a similar result, using timers, without simultaneity, is much more complex.

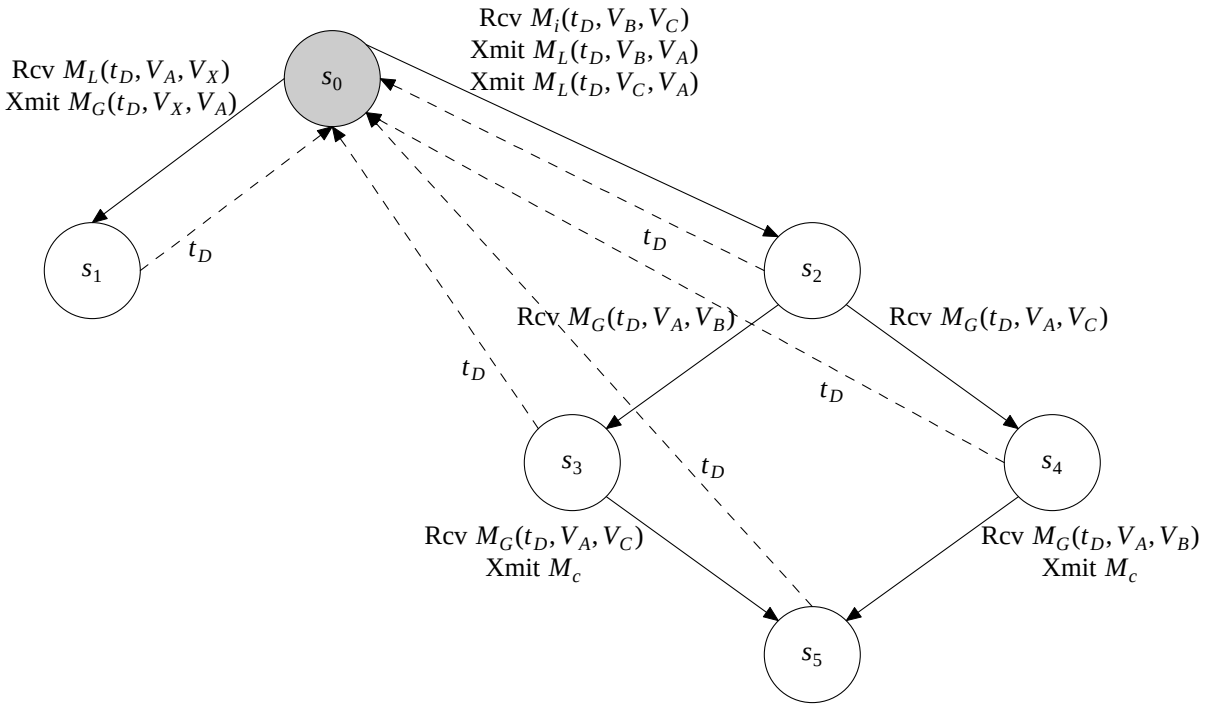


Figure 3. A Simple Lock Protocol

State s_0 is the idle state. If vehicle, V_A in state s_0 receives message, $M_i(t_D, V_B, V_C)$, the higher level merge routine is requesting a lock that lasts until time t_D. V_A requests the lock from vehicles V_B and V_C, by sending messages $M_L(t_D, V_B, V_A)$ and $M_L(t_D, V_C, V_A)$. Once V_A requests the lock it does not return to s_0 until t_D.

If vehicle V_B is in s_0 when it receives $M_L(t_D, V_B, V_A)$, it sends $M_G(t_D, V_A, V_B)$ to indicate that it grants the lock and remains in state s_1 until t_D. Similarly, V_C grants the lock if it is in s_0.

The lock protocol communicates between vehicles over an unreliable communications channel, using the communications stack. In this example it use a point-to-point ARQ protocol that attempts to deliver a message N times before giving up. The messages $M_L()$ and $M_G()$ may not be received. If V_A receives both grant message it sends message M_c to the merge process so that the merge can proceed.

The state machine, as drawn is not complete. A completely specified FSM must have a transition for every input in every state. The complete machine has a self-loop, with a null output, at each state, for any of the messages that aren't included in the figure.

This lock protocol meets the requirements, but is inefficient, in that it may hold the locks longer than necessary and delay other merges. The efficiency can be improved by including messages that indicate when the lock isn't available, as well as when it is granted. It can also be more efficient by modifying requirement 2 to guarantee that none of the locked cars release the lock before any others, *while the merge is in progress.* Once the merge has succeeded or has been aborted, any or all of the participants can release the lock.

Home Work Due in class February 11th

Perform a first level read of papers [4, 13, 26]. All of the papers are available in IEEE xplore, which is available through the Columbia library on-line.

Write 1/2 to 3/4 of a page, 12 point font, for each paper. Describe what the paper is about and the most important points.

Use your own words. Do not copy text from the paper. Do not include figures from the paper.

Please work alone. Do not collaborate.

Each paper should take 20-30 minutes

REFERENCES

[1] A. Leon-Garcia, I. Widjaja "Communication Networks: Fundamental Concepts and Key Architectures," McGraw-Hill, 2000.

[2] S. Keshav, **An Engineering Approach to Computer Networking**, Addison-Wesley, 1997.

[3] Shou-pon Lin, Nicholas F. Maxemchuk, "An Architecture for Collaborative Driving Systems," IEEE ICNP 2012, Oct. 30 - Nov. 2, 2012, Austin, Texas.

[4] Bohyun Kim, N. F. Maxemchuk, "A Safe Driver Assisted Merge Protocol," IEEE Systems Conference (SysCon) 2012, 19-22 Mar. 2012, Vancouver, BC, Canada, pp. 1-4.

[5] Proceedings of the 2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communications, San Francisco, Ca. Oct. 23-28, 2012

[6] GPS Tutorial, http://www.trimble.com/gps/index.html

[7] N. Bulusu, J. Heidemann and D. Estrin, "Adaptive Beacon Placement," Proc. 21st Int. Conf. Distrib. Comp. Syst. (ICDCS-21), Phoenix, Apr. 2001.

[8] S. Capkun, M. Hamdi, J. P. Hubaux, "GPS-free positioning in Mobile Ad-hoc Networks," Proc. 34th HICSS, Jan. 2001.

[9] A. Savvides, C.-C. Han, M. B. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," Proc. 7th ACM/IEEE Conf, on Mobile Computing and Networking (MobiCom 2001.

[10] R. Stoleru, J. A. Stankovic, "Probability Grid: A Location Estimation Scheme for Wireless Sensor Networks," Proc. IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 04), Santa Clara, California, Oct. 2004, pp. 430-438.

[11] F, Benbadis, T. Friedman, M. D. de Amorim, S. Fdida, "GPS-free-free positioning system for wireless sensor networks," 2nd IFIP Conf on Wireless and Optical Communications Networks, WOCN 2005, 6-8 March 2005, pp. 541 - 545.

[12] D. L. Mills, "Internet Time Synchronization: The Network Time Protocol," IEEE Trans. on Communications, Vol. 39, pp. 1482-1493, Oct. 1991.

[13] D. L. Mills, "Improved Algorithms for Synchronizing Computer Network Clocks," IEEE/ACM Trans. on Networking, Vol. 3, No. 3, pp. 245-254, June 1995.

[14] R. Alur, "Timed Automata," 11th International Conference on Computer-Aided Verification, LNCS 1633, pp. 8-22, Springer-Verlag, 1999.

[15] A. Banerjea, D. Ferrari, B. A. Mah, M. Moran, D. C. Verma, H. Zhang, "The TENET Real-Time Protocol Suite: Design, Implementation, and Experisnces," IEEE/ACM Trans. on Networking, Feb. 1966, vol. 4, no. 1, pp. 1-10.

[16] R. Alur, D. L. Dill, "A Theory of Timed Automata," "Theoretical Computer Science," v. 126 pp. 183-235 (1994).

[17] M. Yannakakis, D. Lee, "An Efficient Algorithm for Minimizing Real-Time Transistion Systems," Formal Methods in System Design 11, 1997, pp 113-136, Kluwer Academic Publishers.

[18] J. Park, R. Miller, "Synthesizing protocol specifications from service specifications in timed extended finite state machines," Proc. - Int. Conf. on Distributed Computing Systems, May 27-30, 1997, pp. 253-260.

[19] P. Sinha, N. Suri, "On the use of formal techniques for analyzing dependable real-time protocols," proc. of IEEE Real-Time Systems Symposium, Dec. 1-3, 1999, pp. 126-135.

[20] F. Lin, M. Liu, C. Graff, "On the verification of time-dependent protocols using timed reachability analysis," Proc. of Conf. on System Sciences, Jan. 3-6, 1989, pp. 285-294.

[21] S. Nagano, Y. Hatakeyama, Y. Kakuda, T. Kikuno, "Timed reachability analysis method for EFSM-based communication protocols and its experimental evaluation," Proc. of ICNP, Oct. 29- Nov 1, 1996, pp. 92-96

[22] J.-C. Park, R. E. Miller, "A Compositional Approach For Designing Multifunction Time-dependent Protocols," Proc. of ICNP, Oct. 28-31, pp. 105-112.

[23] C. Huang, S. Lee, J. Hsu, "Probabilistic timed protocol verification for the extended state transition model," Proc. of Int. Conf. on Parallel and Distributed Systems, Dec 19-21, 1994, pp. 432-7.

[24] C. Huang, J. Hsu, S. Lee, "Probabilistic fuzzy timed protocol verification," Computer Communications v 19 n 5 May 1996. p 407-425.

[25] M. A. Fecko, M. U. Uyar, A. Y. Duale, P. D. Amer, P.D.; "A technique to generate feasible tests for communications systems with multiple timers," IEEE/ACM Trans. on Networking, vol. 11, iss. 5, 2003, pp. 796-809.

[26] Patcharinee Tientrakool, Ya-Chi Ho, N. F. Maxemchuk, "Highway Capacity Advantage from Using Vehicle-to-Vehicle Communication Sensors for Collision Avoidance," Proceedings of 2011 IEEE Vehicular Technology Conference, San Francisco, Sept. 5-6, 2011, Pgs. 1-5.