Networking is IPC

Step 1: Getting two systems to Communicate
Reliable Transfer of Data

John Day

2014

"A good (network) Architect suffers
from the topologist's vision defect.
He can't tell a coffee cup from a doughnut."

Getting from A to B Reliably

- We know:
    - Bad things can happen to messages in transit. Protection against lost or corrupted messages. Error Control
        - This can be expensive
    - Receiver must be able to tell sender, it is going too fast. We need Flow Control, and
    - We have lost our means of synchronization:
        - No common test and set means shared memory can no longer be used
        - Must create shared state between two systems. An explicit synchronization mechanism is required.
- So, we need some kind of Protocol for Error and Flow Control.

All Communication goes through Three Phases

Enrollment

Create sufficient state within the network to allow an instance of communication to be created.

We tend to gloss over or ignore this one. But less so recently.

Allocation (Establishment)

Creating sufficient shared state among instances to support the functions of the data transfer phase.

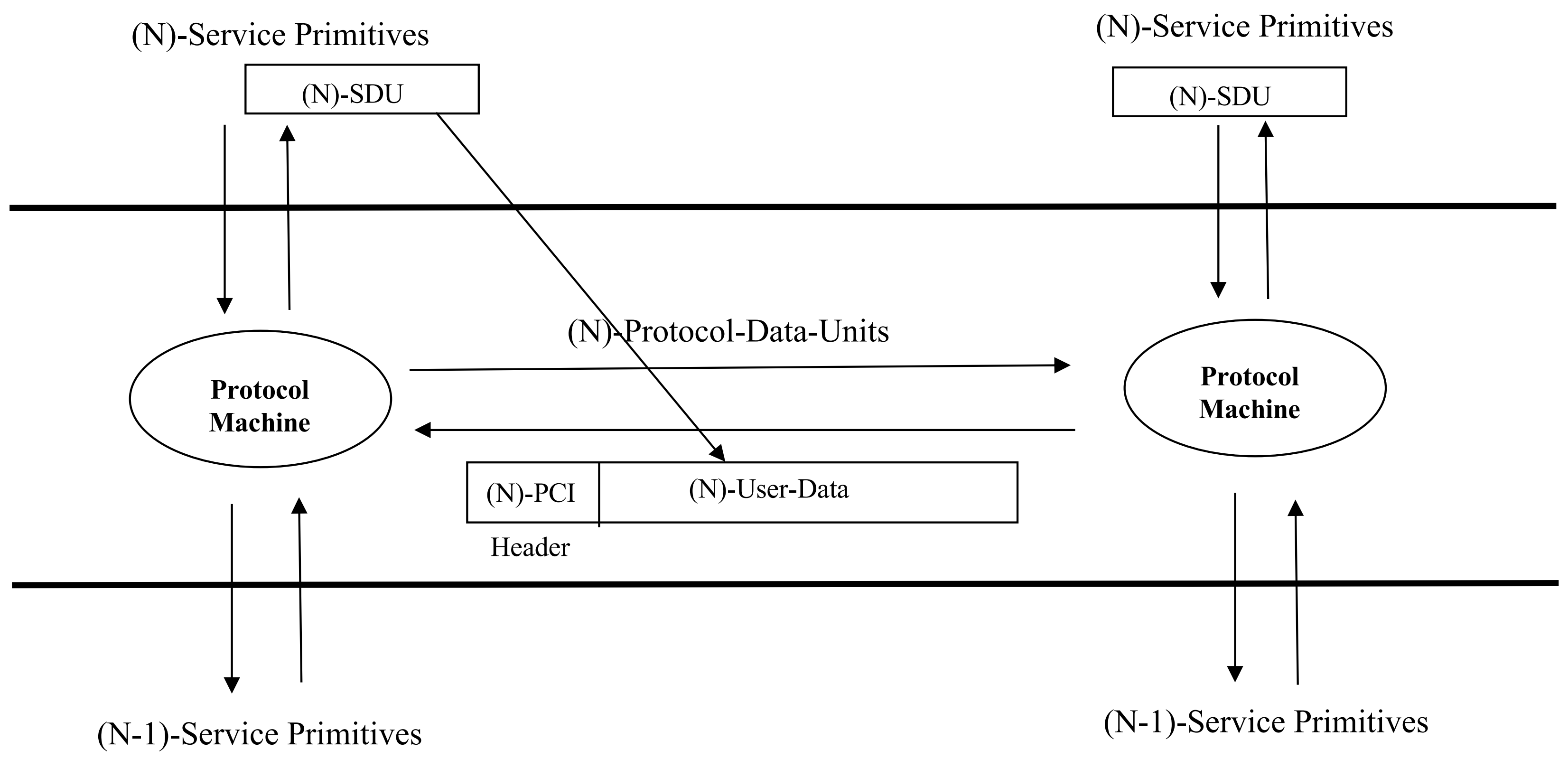
Data Transfer

Provide the actual transfer of data and functions to support it. (Application protocols may introduce subphases.)

Two Elements of Every Design

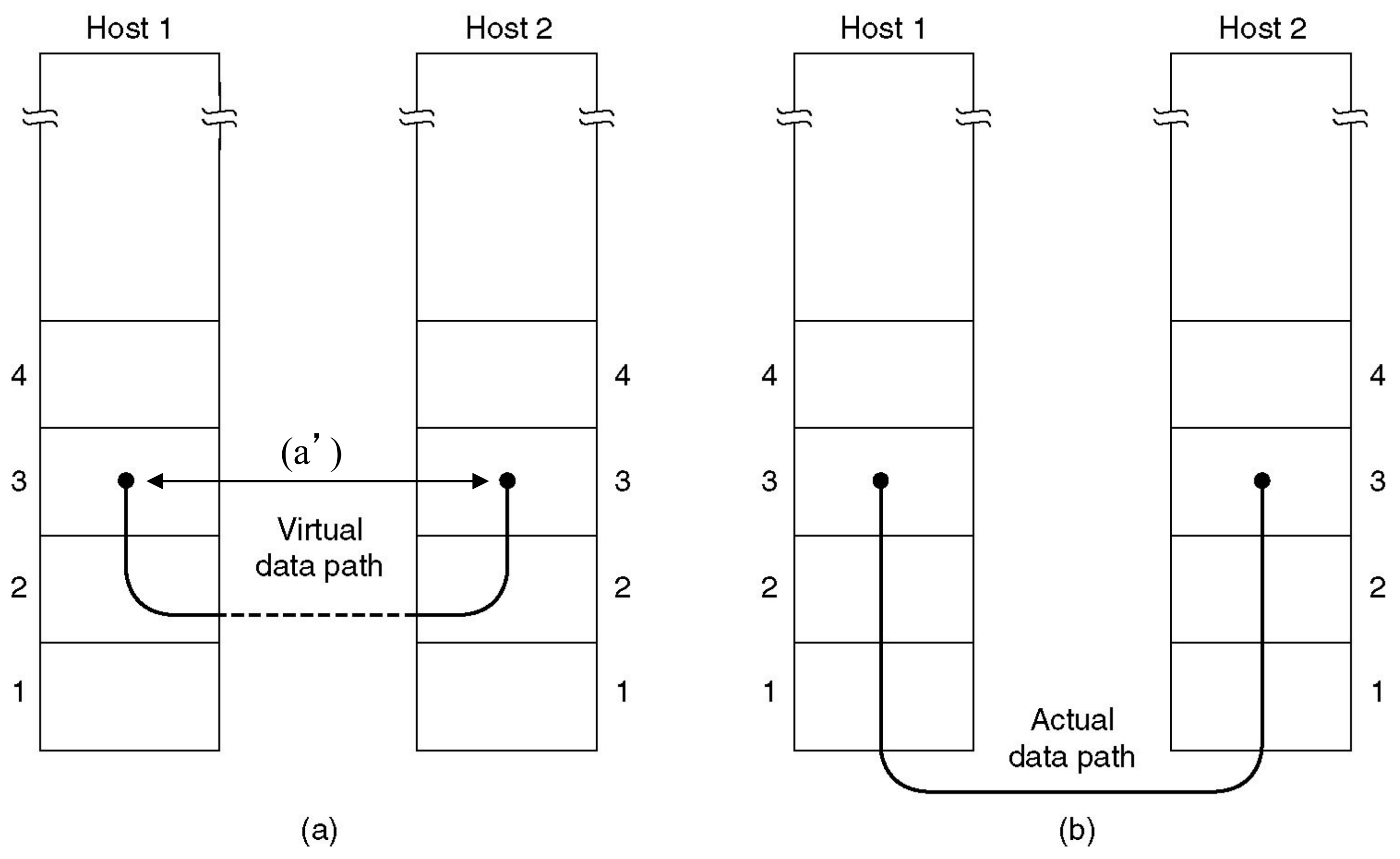
- Service/Interface Provided to the Layer Above
  - The black box view. Abstracts and hides the internal workings of the protocol.
  - Services are those characteristics observable by the user of the protocol.
- Protocol Definition
  - Finite state machine specifying the behavior of the protocol.
  - Driven by service/interface primitives
  - Functions implement the services. Their action may or may not be visible to the user of the layer.
- Minimal Service Required from the Lower Layer
  - If a new lower layer doesn’t meet this level of service, then additional functionality is required to make up the difference.

Protocol Operation



PDUs are also known as messages, packets, segments, frames, cells, etc.
While not rigid, the terms are generally used with specific layers.
PDU stresses that they are really all the same thing.
(SDU = Service Data Unit)

Virtual and Actual Flow



(a) Virtual communication. (T) View not logically consistent.

(a') Virtual communication. (D)

(b) Actual communication.

For communication to occur there must be shared state and a protocol at the layer above even if it is an application.

Two Kinds of Protocol

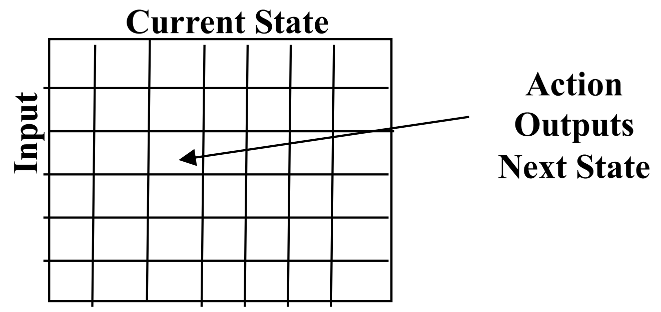
- Two kinds of Protocol
  - Application Protocols
  - Data Transfer Protocols
- Both are concerned with establishing and maintaining shared state.
  - Application Protocols modify state *external* to the protocol.
  - Data Transfer Protocols modify state *internal* to the protocol.

Services of an IPC (data transfer) Protocol

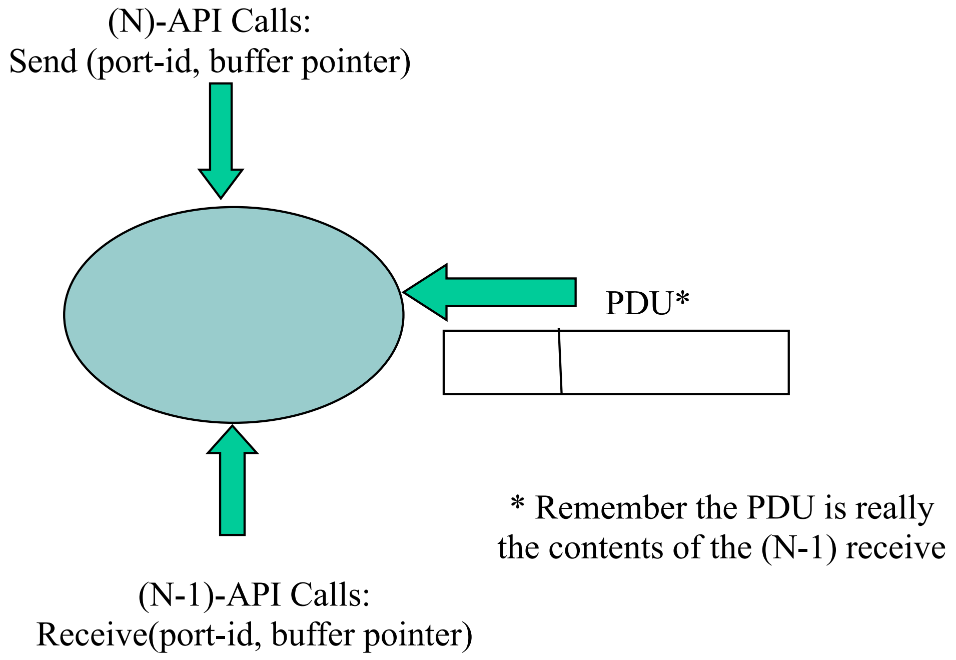
- Allocation of Communication Resources
  - Listen, Connect, Disconnect, etc.
- Send/Receive SDUs
- Stream vs Record vs Idempotent
- Quality of Service
- Contrary to T, connection/connectionless is not a service but a function.
    - We will return to this in a few weeks
  - Should not be visible across the layer boundary.
    - Even OSI called it a mode, whatever that is.

What Does an Implementation Look Like?

- Protocol Machines get inputs from above, below and to the side.
- In general, each function uses information in the PCI (the header)
  - But this is overhead, so want to keep it small.
- The PCI is used to update and maintain the distributed shared state.
- The Implementation is best represented as a Finite State Machine.
  - Can't block on one interface. Input can occur at any time, any where.



State Table


(N)-API Calls:
Send (port-id, buffer pointer)
PDU*
(N-1)-API Calls:
Receive(port-id, buffer pointer)

* Remember the PDU is really the contents of the (N-1) receive

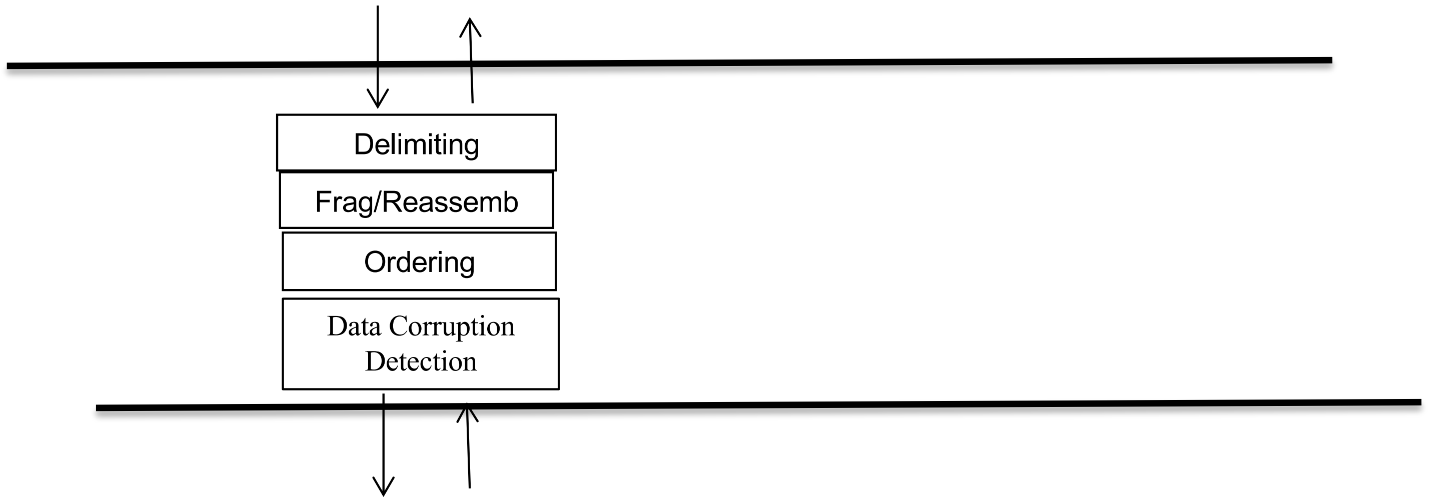
Functions of a Protocol: (1)

- For roughly 50 years, we have been designing data communication protocols.
- In that time a small set of functions have been found to be useful. No new functions have appeared for decades.
- Every OS textbook stresses the importance of separating mechanism and policy.
- Networking textbooks never mention it.
- We are seeing the same mechanisms with different policies.
  - Separating mechanism and policy is going to prove important.

- First, we will look at these functions and then consider how to organize them into a protocol.

Functions of a Protocol (2)

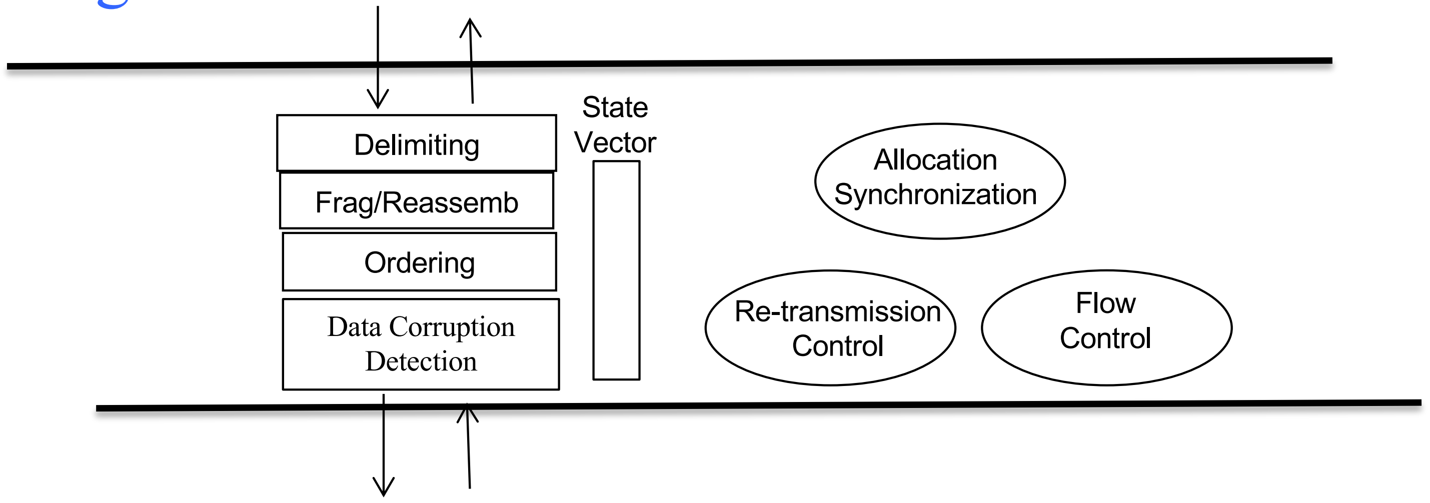
Our Legos: I



- **Data Corruption Detection/Correction** – Making sure that the bits in the packet are correct (within some small probability)
- **Delimiting** – Packing SDUs into the User Data of PDUs and Marking the boundaries of an SDU, so we can find them again.
- **Fragmentation/Reassembly; Combining/Separation** – Last Step of Delimiting, Creating User-Data fields of the PDUs
- **Ordering/Sequencing** – Putting sequence numbers on PDUs when sending them so we can keep them in order when they arrive.
- **Lost and Duplicate Detection** – Detecting entire lost PDUs, or duplicates due to retransmission.

Functions of a Protocol (3)

Our Legos: II



- **Retransmission Control (Acknowledgement)** – Recovering from lost PDUs.
- **Flow Control** – Keeping the sender from overrunning the receiver, using either a Sliding Window or Rate-based.
- These two are feedback mechanisms, therefore we need:
- **Synchronization** – Ensuring that the sender and receiver maintain consistent state relative to each other.
- **Allocation/Deallocation** – Allocating/Deallocating ports and creating and terminating synchronization.

Data Corruption Detection and Correction

(The first function to apply to an inbound SDU and the last function to apply to an outbound SDU)

- Error-Detecting Codes
  - Only enough to know there is a problem

- Error-Correcting Codes
  - Enough redundancy to fix the problem

Error Codes (1)

1. Hamming codes.
2. Binary convolutional codes.
3. Reed-Solomon codes.
4. Low-Density Parity Check codes.

Error Detection Codes (1)

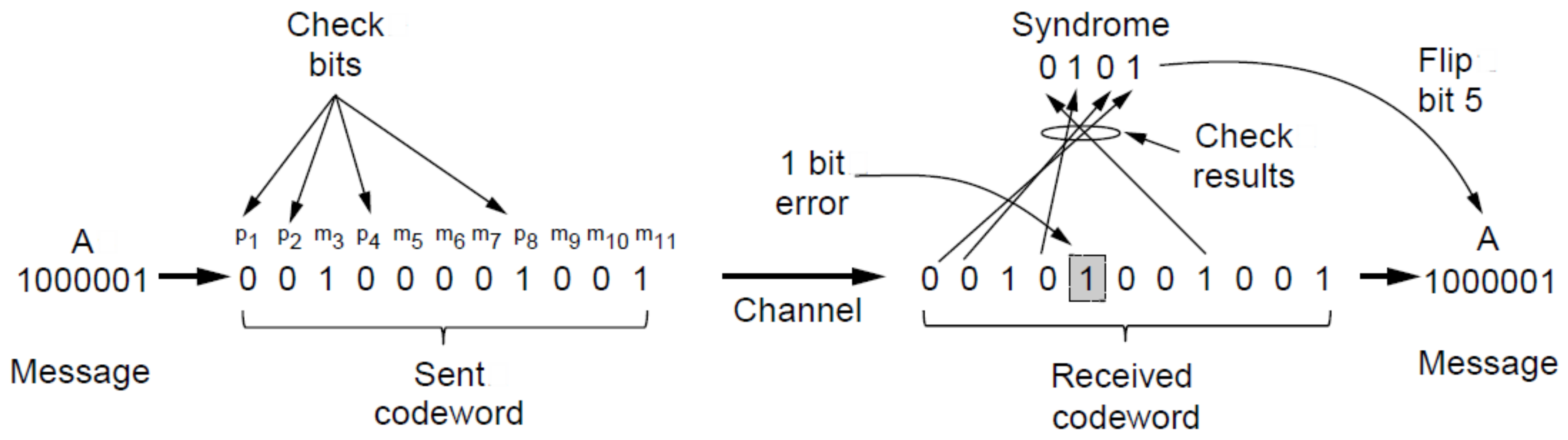
A Frame consists of **m** data bits and **r** check bits for a total of **n** bits

```
10001001
10110001
--------
00111000
```

- The number of bits that two codewords differ or the number of bit flips to transform one codeword to another is the Hamming distance.
- To detect d errors (d+1) bits are required
- To correct d errors (2d + 1) bits are required.
- To correct all single bit errors. For **m** message bits and **r** check bits, we need:

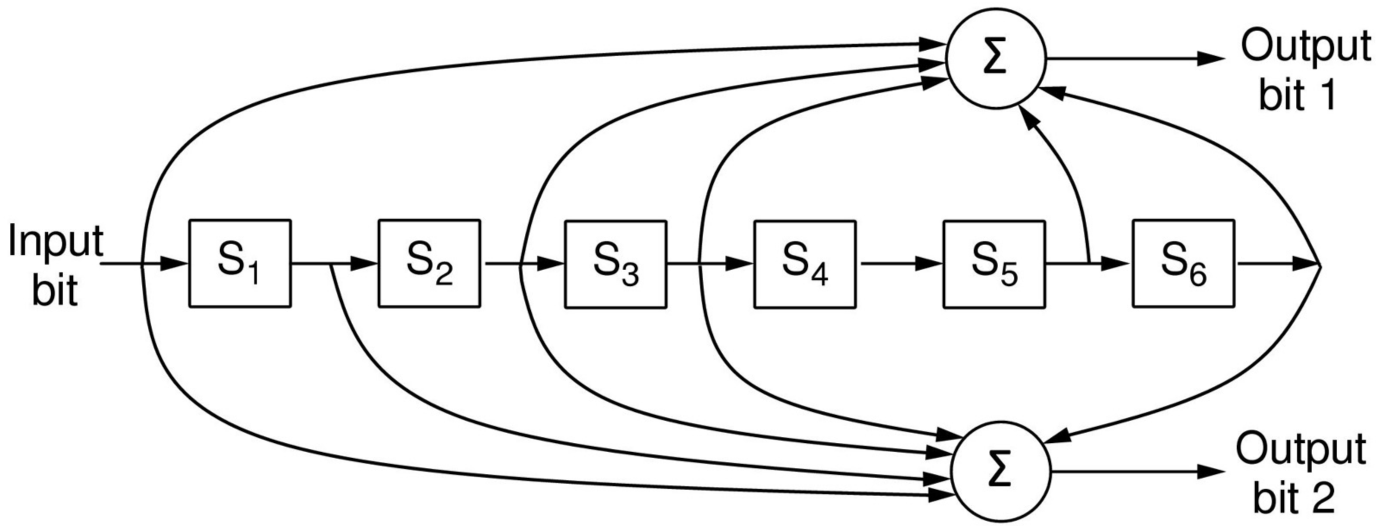
$$m + r + 1 \leq 2^r$$

Error Detection Codes (2)



Example of an (11, 7) Hamming code correcting a single-bit error.

Error Detection Codes (3)



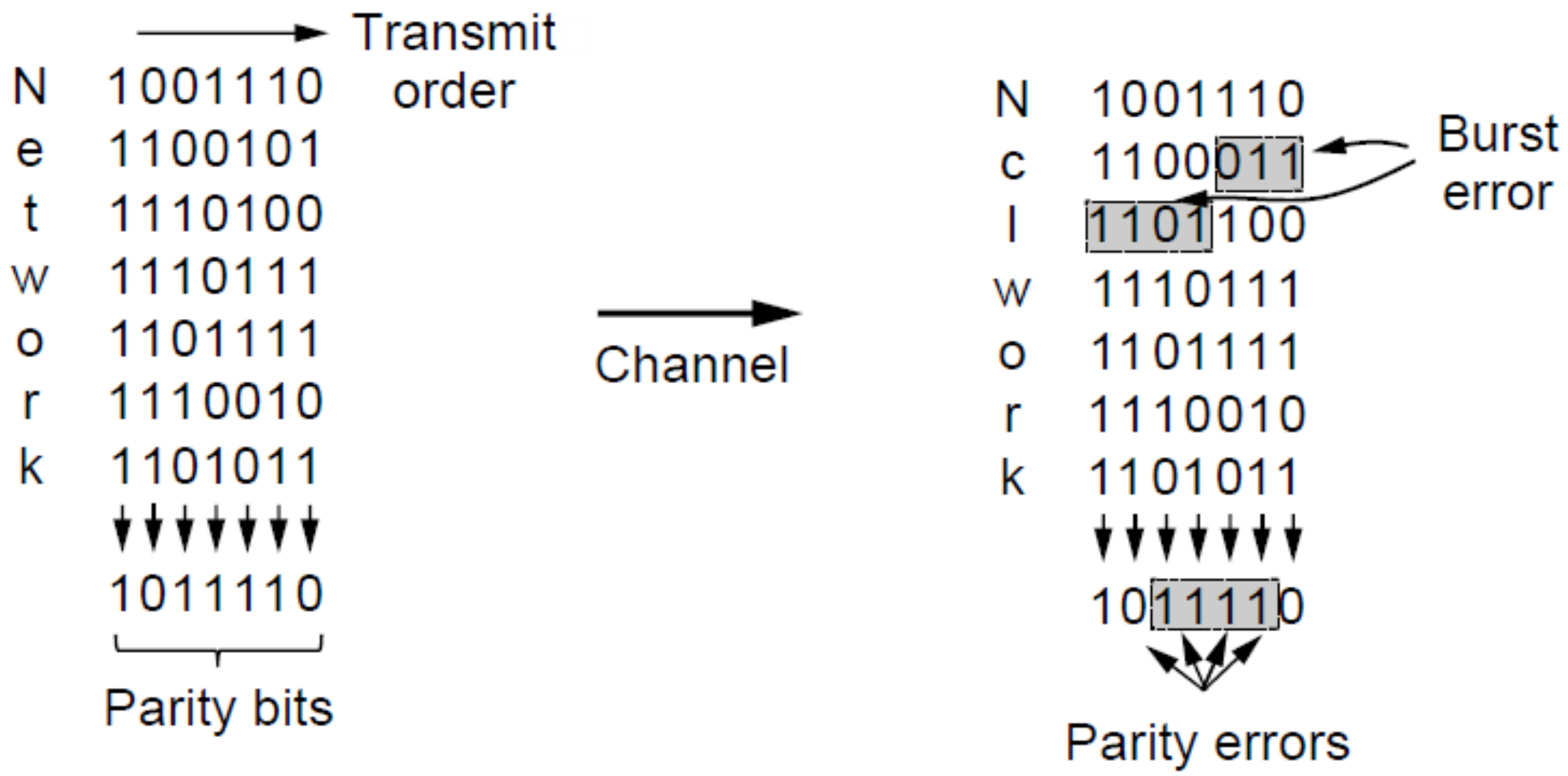
The NASA binary convolutional code used in 802.11.

Error-Detecting Codes (4)

Linear, systematic block codes

1. Parity.
2. Checksums.
3. Cyclic Redundancy Checks (CRCs).

Error-Detecting Codes (5)



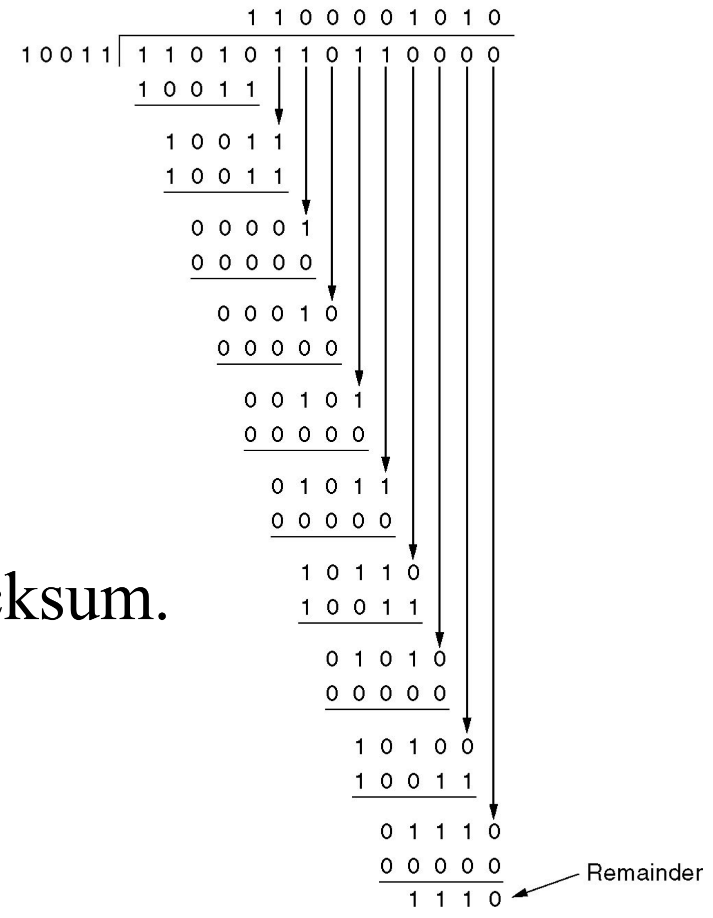
Interleaving of parity bits to detect a burst error.

Error-Detecting Codes (6)

Frame : 1 1 0 1 0 1 1 0 1 1

Generator: 1 0 0 1 1

Message after 4 zero bits are appended: 1 1 0 1 0 1 1 0 1 1 0 0 0 0

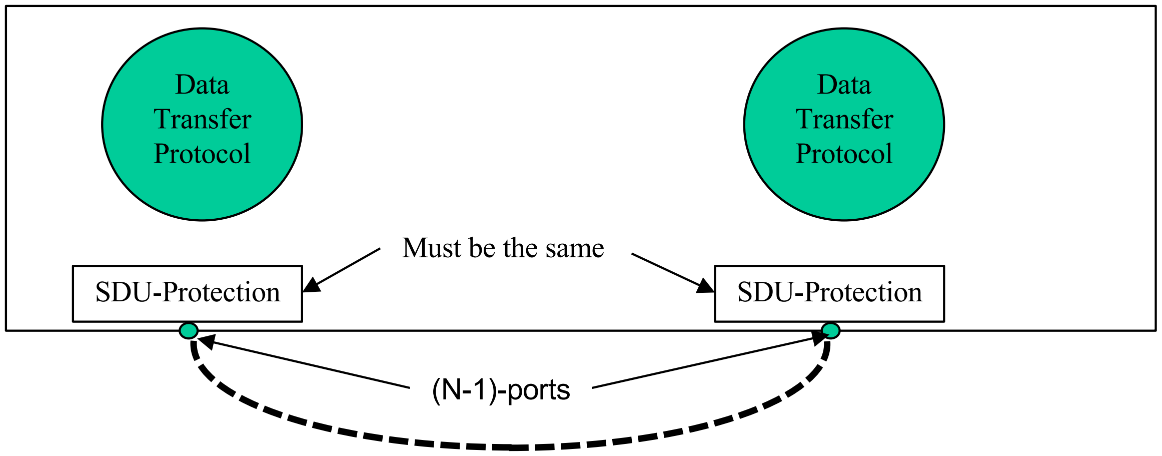


Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

Calculation of the polynomial code checksum.

Data Corruption Detection Summary: I

- Error Codes are chosen based on the characteristics of the lower layer and desired PDU size: burst errors, single bit errors, etc.
- Notice the (N-1)-port the PDU is delivered to must use the same Error Corruption policy as the (N-1)-port the PDU was sent on.
  - Hence, strictly speaking it is not part of the protocol. Any protocol using the (N-1)-port will use the same error policy.
- IOW,



Therefore, SDU Protection doesn't need to be part of the protocol.
(Later, we will add compression, encryption, etc. to SDU Protection.)

Data Corruption Detection Summary: II

- Common way to compute CRC in software is by table look-up.
- Beware bit stuffing. Undermines the CRC.
- Error codes are one of those important details like numerical analysis that we like to ignore. Do so at your peril.
- Adds a 1, 2, or 4-bytes field to the PCI for the error code, some will have one for the header only, the whole message, or both.
  - In older protocols, it may be in a trailer.
- Mechanism: Apply the Error Detection Policy to the PDU
  - Policy: Choice of Error Detection Scheme, e.g., error polynomial.
  - No State to Record: PDU is okay, or if it isn't, discard the PDU.

Delimiting

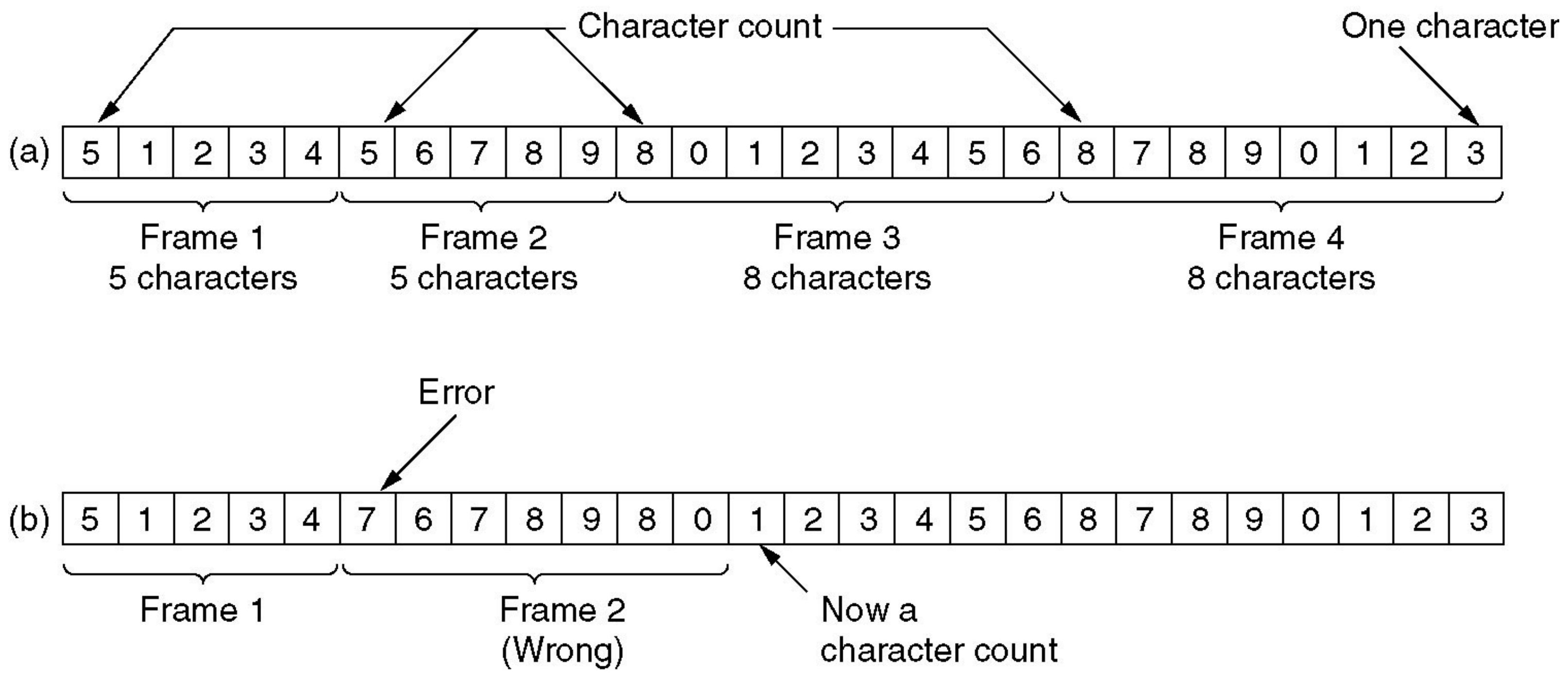
Determining the Bounds of a SDU

(The first function to apply to an outbound SDU and the last function to apply to an inbound SDU)

- Two Parts to Delimiting:
- Bounding the SDU
  - Internal Delimiting
    - Length field in the PCI
  - External Delimiting
    - Byte level Flag with byte stuffing
    - Bit level Flag with bit stuffing
    - Bit level encoding
- Packaging SDUs into the User Data field of PDUs
  - Fragmentation/Reassembly
  - Concatenation/Separation

Bounding SDUs (1)

Internal Delimiting



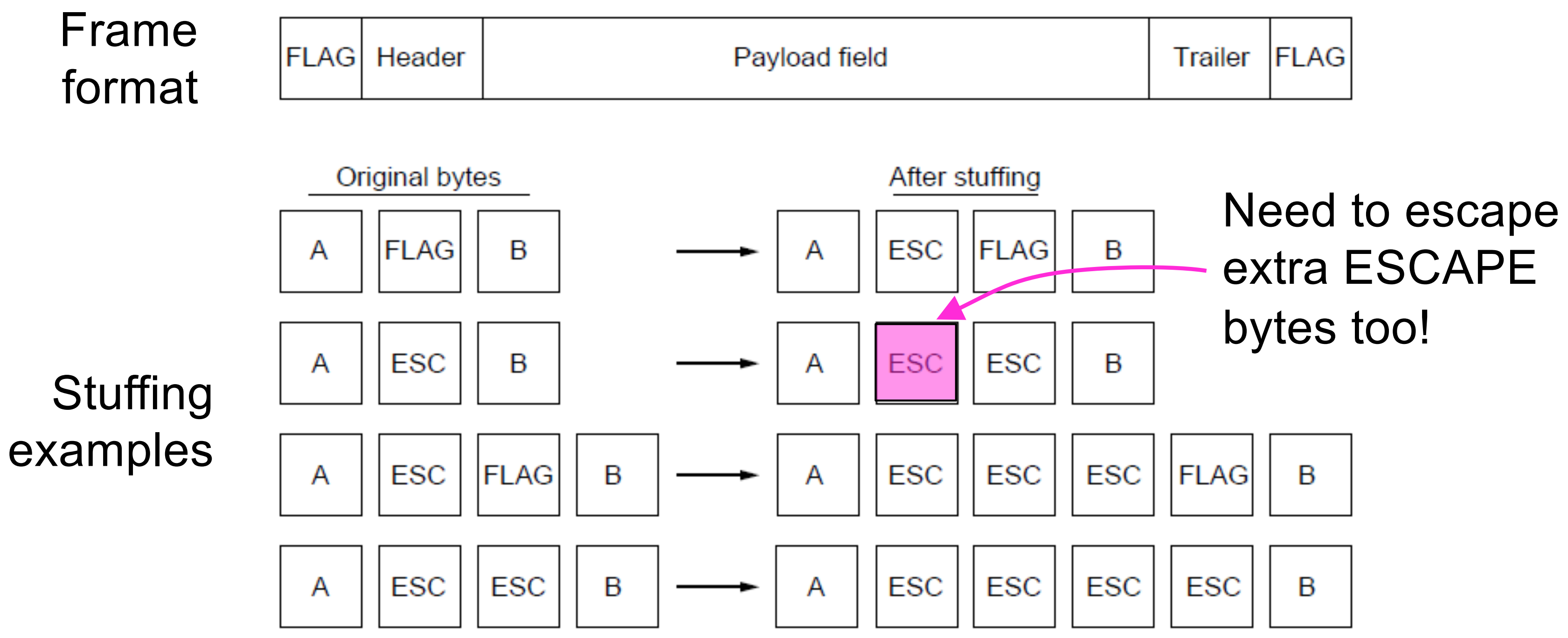
A character stream. (a) Without errors. (b) With one error.

Bounding SDUs (2)

External Delimiting

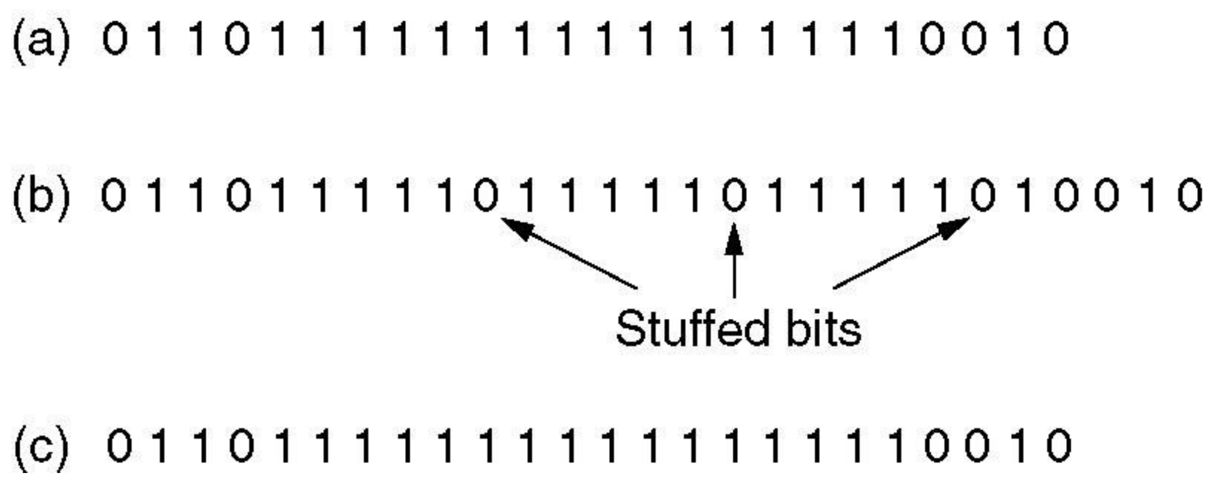
Special flag bytes delimit frames; occurrences of flags in the data must be stuffed (escaped)

- Longer, but easy to resynchronize after error



Delimiting (3)

External Delimiting


(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0
(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0
Stuffed bits
(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Bit stuffing

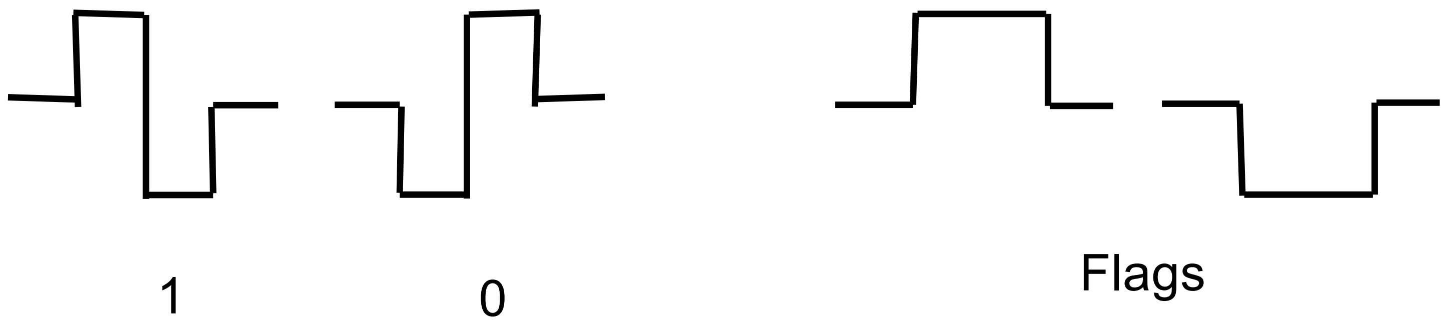
(a) The original data.

(b) The data as they appear on the line.

(c) The data as they are stored in receiver's memory after unstuffing.

Delimiting (4)

External Delimiting



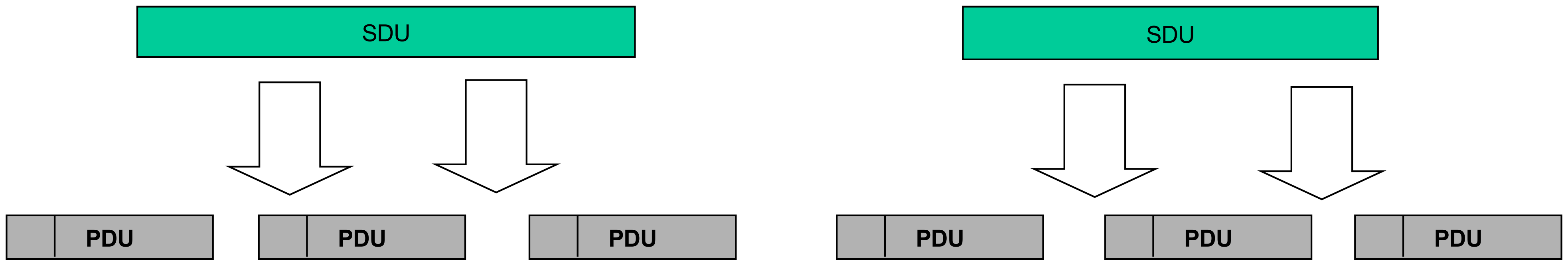
Bit encoding

Manchester Encoding (LANs)

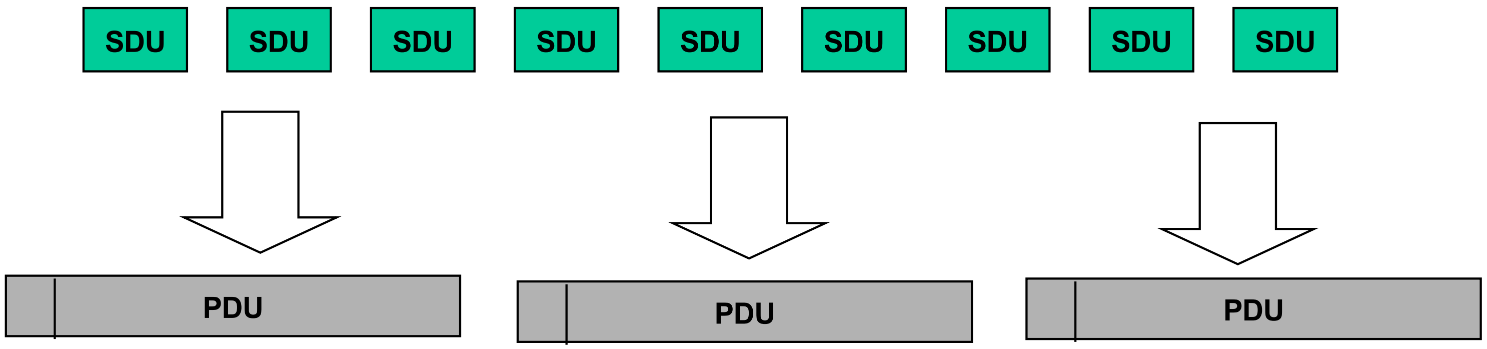
(Notice this works by encoding the delimiting flags in a different space, a different alphabet, (analog) orthogonal to the space for the data (digital).)

Packaging Functions (1)a

- Fragmentation/Reassembly
  - Big SDUs into smaller PDUs
    - In a well-formed architecture, Max PDU size is a property of the Layer, so
  - To fit buffer constraints of hardware or more likely error probability, i.e. the longer a PDU the more likely an error and hence a re-transmission.
  - Generally, encoded with the Sequence Number and a More Bit.
    - More on this later.
- Adds to the PCI a More or Last Fragment bit to indicate last Fragment, and a field to place the fragment within the SDU, e.g., count or offset.



Packaging Functions (2)



- Combining/Separation - Little SDUs into a PDU.
  - Done for transmission efficiency, but must be concerned about delay.
    - Fewer PDUs to look at in the relaying.
  - Implementation is basically a delimiting problem.
    - Length fields preferred. Possibly a sequence of lengths pointing to each concatenated SDU.
  - Not Done in IP Networks mainly because of historical reasons and delay (low density traffic). An artifact of not pursuing connectionless.
  - Two methods: 1) chain of <length, SDU>
  - or 2) table of pointers into the data: <offset 1, . . , offset n><SDU 1><SDU 2> . . .<SDU n>

Delimiting Policies

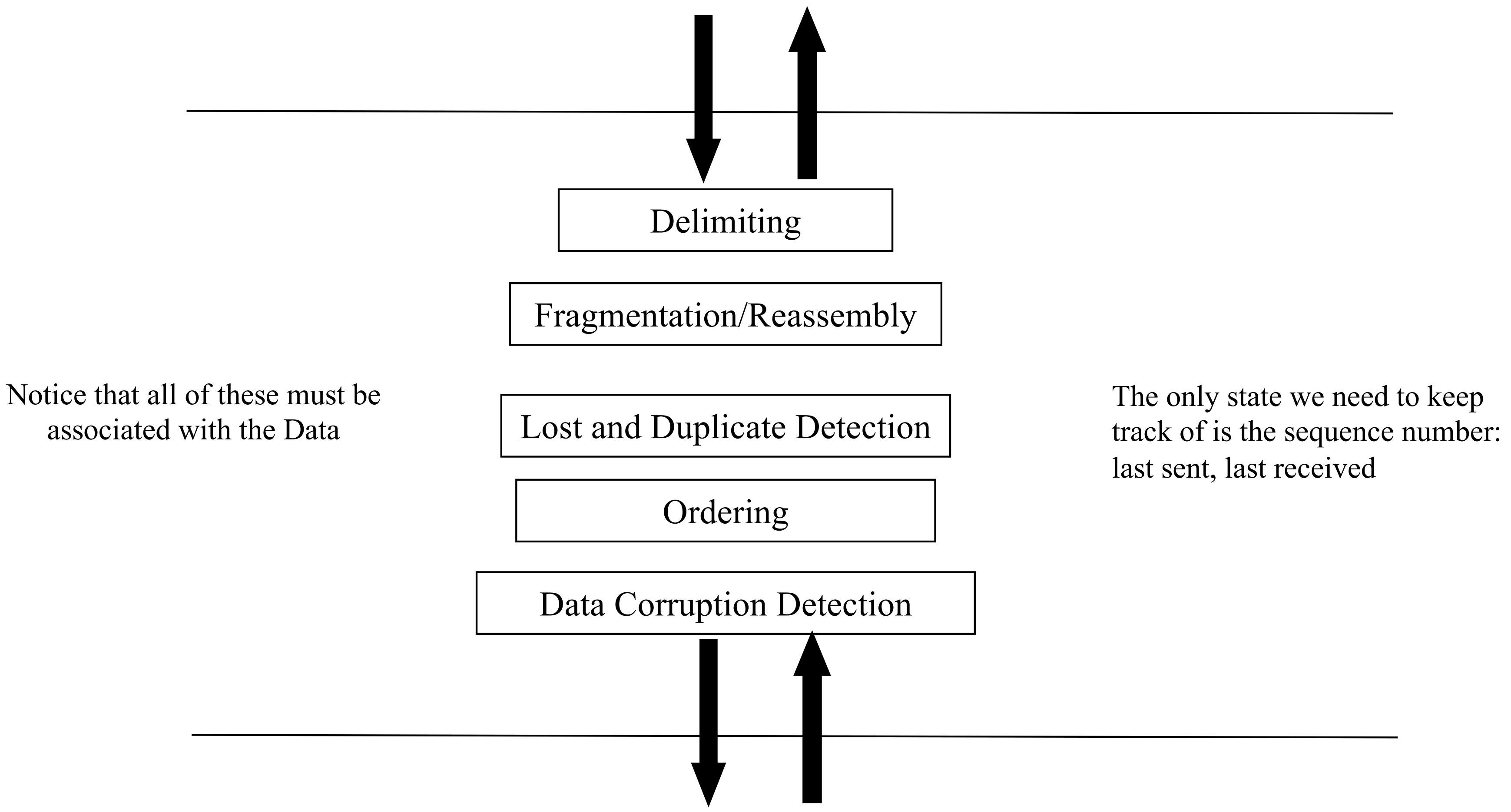
- Determine Maximum Transmission Unit (MTU), e.g., Max PDU size
- At design time, chose internal or external delimiting. External for a media layer, otherwise use internal delimiting, e.g. length field.
- What will typical SDU sizes be? If (N+1) is a Layer, it will be (N+1)-MTU; otherwise, it will be a range.
- Determine criteria for fragmentation or concatenation.
  - Mostly determined at design time.
- No state to record, self-correcting.
- We now have the means to put data into a PDU, so we can send it and deliver the same bytes.

More Useful Mechanisms (3)

- Ordering
  - Use Sequence numbers
    - Must be careful about initial value and roll-over.
    - Byte sequencing vs PDU sequencing
    - Important that retransmissions are on the same boundaries.
  - Some Applications may use more complex ordering schemes relative to multiple flows.
    - Partial ordering, total ordering, etc.
- Lost and Duplicate Detection
  - Use the Sequence Numbers to detect duplicates and gaps
  - Are Gaps Allowed? How many SDUs?

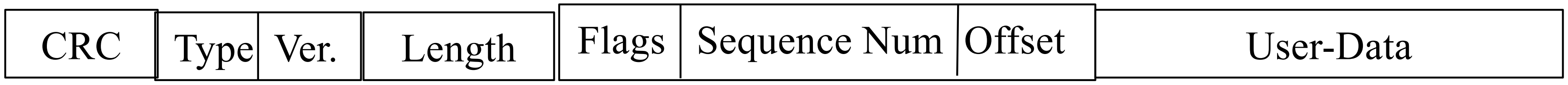
Data Transfer Protocol

What We Have So Far


Delimiting
Fragmentation/Reassembly
Lost and Duplicate Detection
Ordering
Data Corruption Detection
Notice that all of these must be associated with the Data
The only state we need to keep track of is the sequence number: last sent, last received

- SDUs must be delimited when they appear so that we can maintain their integrity.
  - That means that the last thing done to them before they come to us from below is delimiting as well.

What Might a PDU Look Like?
(so far)



- First, the CRC to ensure the bits are not corrupted.
  - Must be in a fixed place, either first or a fixed offset
- Then, the usual things we need:
  - Type of PDU, and Version of the Protocol, Length
- Flags (More Frag) and Sequence Number for Ordering, and perhaps an Offset to Support Fragmentation
- O, and the User Data.
- Notice the fixed length fields facilitate parsing.

More Useful Mechanisms (4)

- Retransmission Control (really)
  - Acknowledgement
  - Negative Acknowledgement
- Flow Control
  - Start-Stop
  - Credit-based
    - Fixed Window
    - Dynamic Window
  - Rate-based (Timer-based)
- Synchronization/Allocation
- Reset - Force receiving state machine to a known state.
  - As we will see, can create issues.

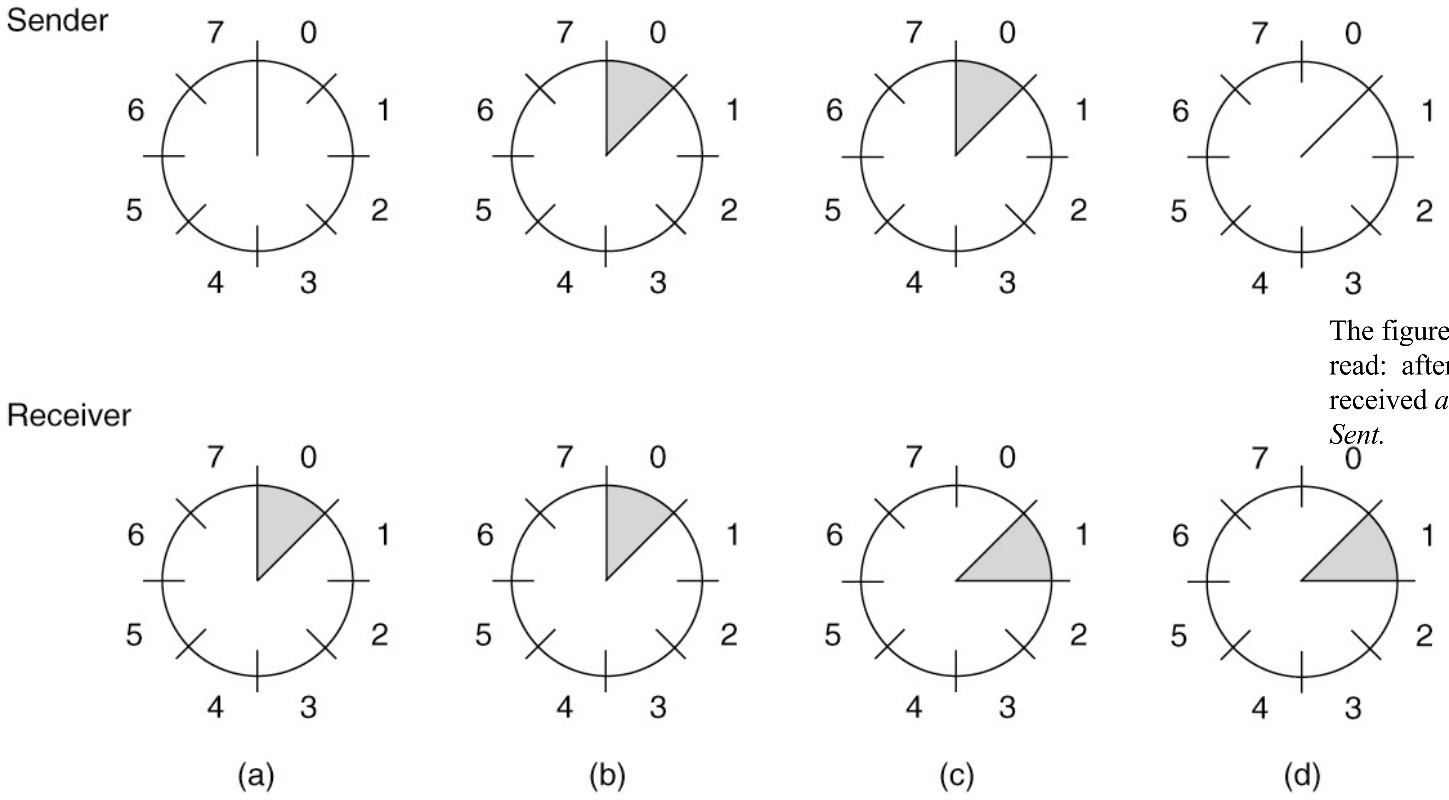
Bidirectional Transmission, Multiple Frames in Flight (1 of 3) (traditional view)

- Bidirectional transmission: piggybacking
  - Use the same link for data in both directions
  - Interleave data and control frames on the same link
  - Temporarily delay outgoing acknowledgements so they can be hooked onto the next outgoing data frame
- Piggybacking advantages
  - A better use of the available channel bandwidth
  - Lighter processing load at the receiver
- Piggybacking disadvantages
  - Determining time data link layer waits for a packet to piggyback the acknowledgement.
  - If the traffic isn’t symmetrical, no point.
  - If the reverse traffic is more than a few 10s of bytes, savings is negligible.

Bidirectional Transmission, Multiple Frames in Flight (2 of 3) (traditional view)

- Three bidirectional sliding window protocols
  - One-bit sliding window, go-back-n, selective repeat
    - Differs in how wide the window is. The width is always fixed.
- Consider any instant of time
  - Sender maintains a set of sequence numbers corresponding to frames it is permitted to send
  - Frames are said to fall within the sending window
  - Receiver maintains a receiving window corresponding to the set of frames it is permitted to accept
- Differ among themselves in terms of efficiency, complexity, and buffer requirements

Bidirectional Transmission, Multiple Frames in Flight, (3 of 3) (fixed window)



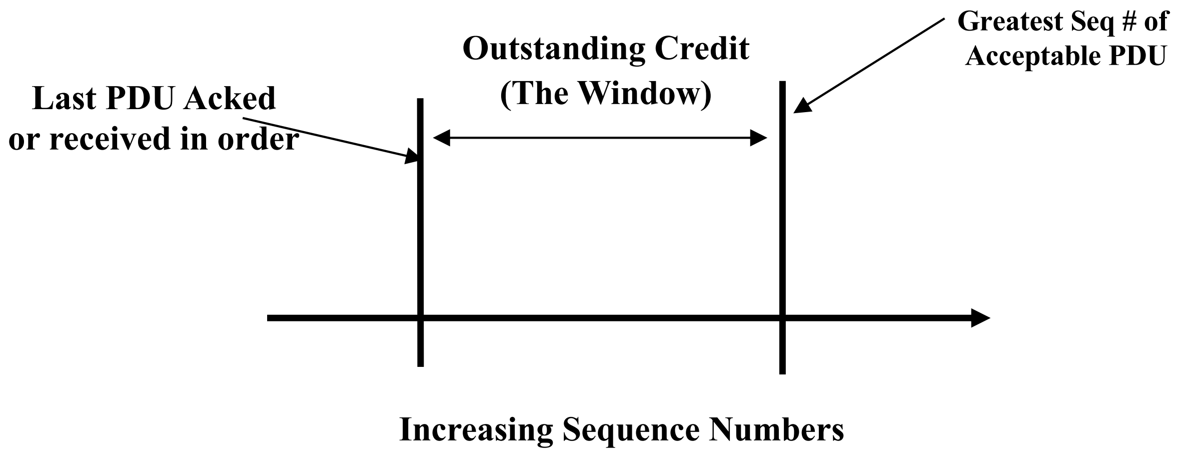
The figure is inaccurate. c) should read: after the first frame has been received *and an Ack has been Sent.*

A sliding window of size 1, with a 3-bit sequence number. (a) Initially. (b) After the first frame has been sent. (c) After the first frame has been received. (d) After the first acknowledgement has been received.

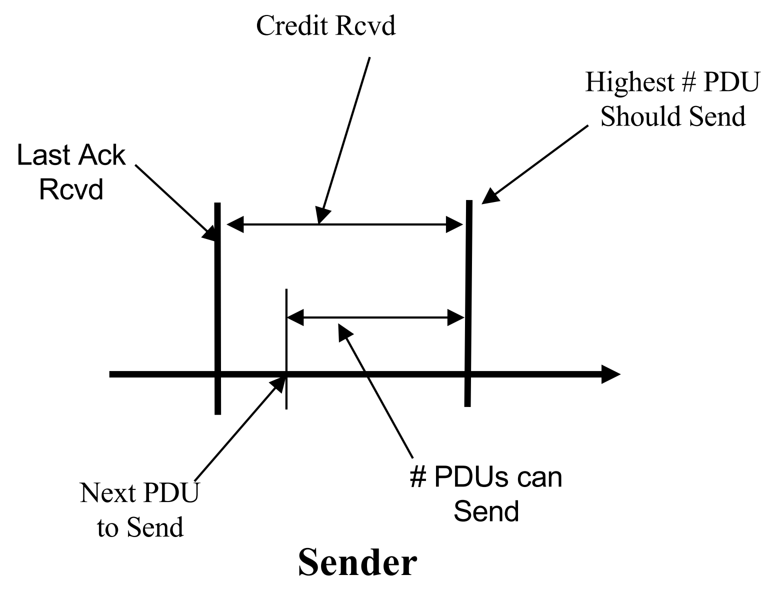
This is a fixed window. Sending Ack advances the window and allows another PDU to be sent.

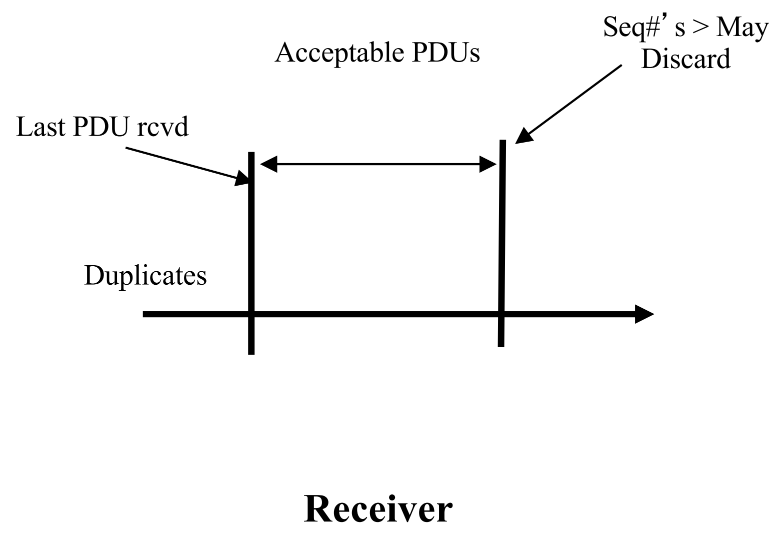
The Sliding Window (the general case)

- As we have seen the operation of several functions are tied to sequence numbers and the sliding window.
- Now allow the window to be a variable width, decouple the ack from the flow control.



Sliding Window Operation (dynamic window)





- The Sliding Window brings together Ordering, Lost and Duplicate Detection, Acknowledgement/Retransmission and Flow Control
  - Pair of windows required for each direction.
  - If no Ack after x, all PDUs in the re-xmsn queue are re-sent.
- Feedback requires synchronization.

Retransmission and Flow Control Policies

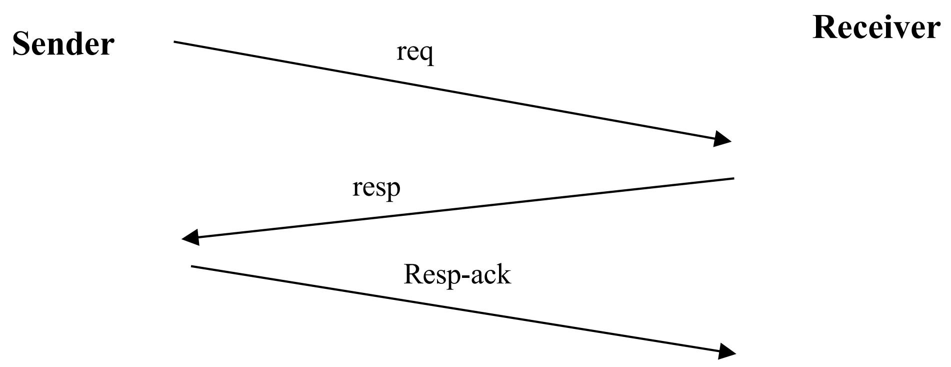
- Retransmission
  - Minimize Retransmissions
  - Estimate Round Trip Time (RTT) for retransmission timeout.
  - Determine the Upper Bound on How Long before Sending Ack, A
  - Algorithm for sending an Ack, e.g., every PDU, every other one, etc.
- Flow Control
  - Goal: Generally, to keep the pipe full or maintain requested data rate without overrunning the receiver.
  - Determine the bandwidth/delay product
  - Pooled vs Static buffers
  - Determine Algorithm for moving right window edge, e.g., try to keep window width constant, fraction of available buffer space, etc.

Synchronization
(Establishment)

- These last two functions are feedback functions. They require more robust shared state.
  - Source and destination must be on the same page.
- What is Synchronized?
  - Initial PCI values,
    - Sequence numbers,
    - initial credit etc.
  - max PDU size,
  - Consistent sequence numbers, credit etc.
- Loose synchronization: The protocol machines are not required to be in the same state but only in a consistent state.

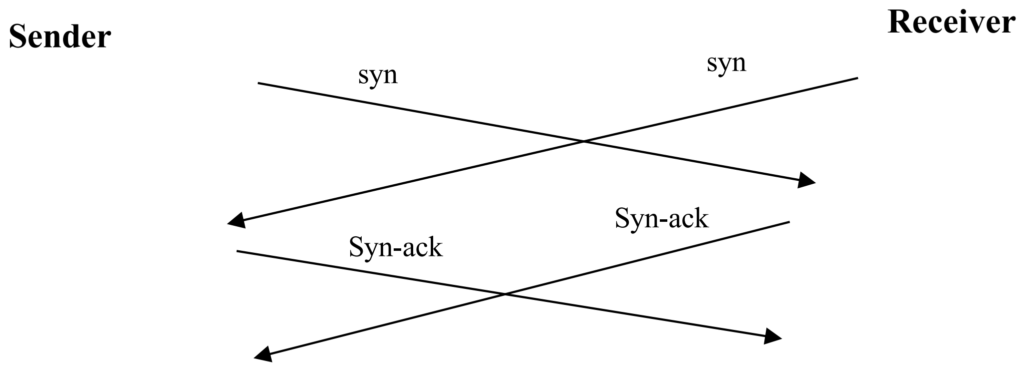
Explicit Synchronization Mechanisms
(So Say the Textbooks)

- No explicit synchronization,
  - i.e., enrollment state is sufficient.
- The 2-way handshake: Request/Response
  - Used when lower layer is very reliable
  - Or when correct operation of the sender is independent of the state of the receiver, i.e., no feedback mechanisms.
- The 3-way handshake used when there is feedback and the potential for lost messages.



Synchronization Details

- What happens when they cross in the mail?
- Depends on the point of view:
  - Asymmetric synchronization
    - As seen by the user of the layer
    - Two flows based on who was the "initiator"
  - Symmetric synchronization
    - As seen by the protocol
    - One flow, each "req" is viewed by the recipient as a "resp"



Disconnect

- Terminating shared state. Not much to it.
- Disconnect/Disconnect Ack or a symmetric Disconnect.

- Later we will see how establishment and disconnect are unnecessary.

3-Way Handshake is a Myth!

- While all of this fits our intuitions,
  - it is totally irrelevant.
- 2-way and 3-way handshakes have nothing to do with *synchronization*.
  - As much as an incantation: Abracadabra! Hocus-Pocus! Alla-Kazam!
- More complexity than the problem requires.

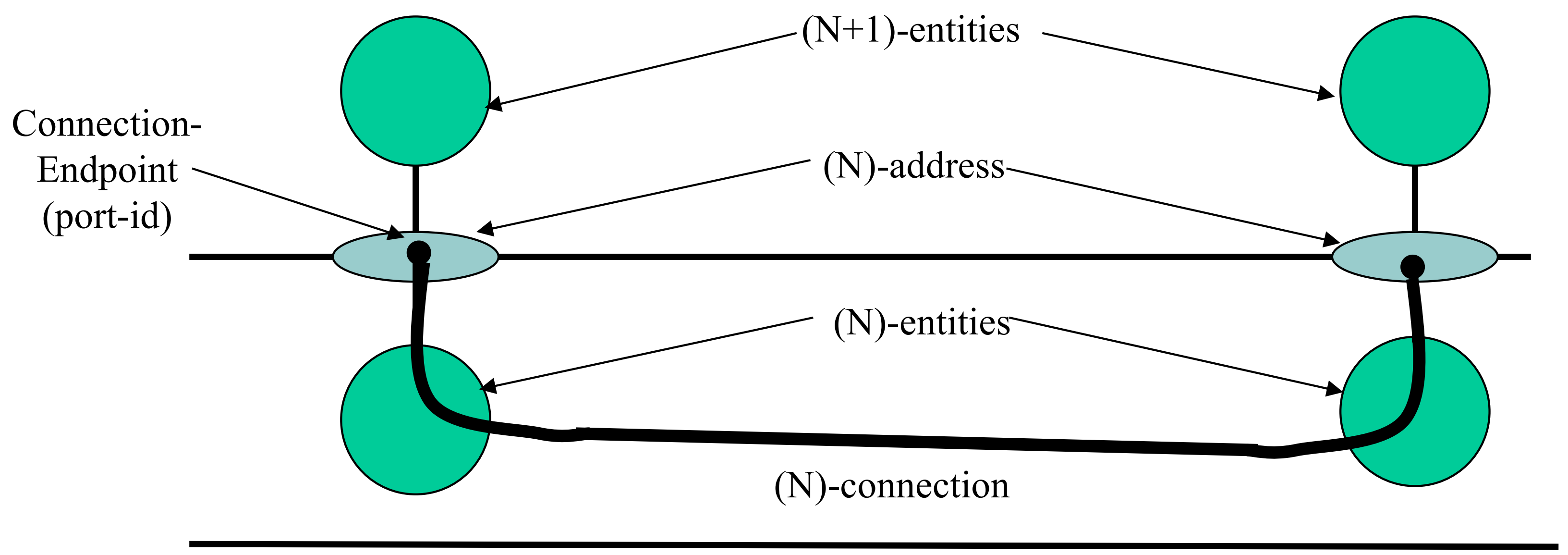
Watson's Theorem

The Fundamental Theorem of Networking

- In the late 1970s, Richard Watson made a remarkable discovery.
- The *necessary and sufficient* conditions for Synchronization for Reliable Data Transfer is to enforce an upper bound on three times:
  - MPL - Maximum Packet Lifetime
  - A - Maximum Hold Time before Ack
  - R - Maximum Time to Re-try
- Assumes all connections exist all the time and always have
- A state vector is merely a cache of information on 'recently active' connections, where 'recently active' means:
- No traffic for 2(MPL+A+R), discard the state vector.
- 3 PDUs *are* Exchanged (lots more than that), but that isn't why it works.
- It works Because, the Times are Bounded

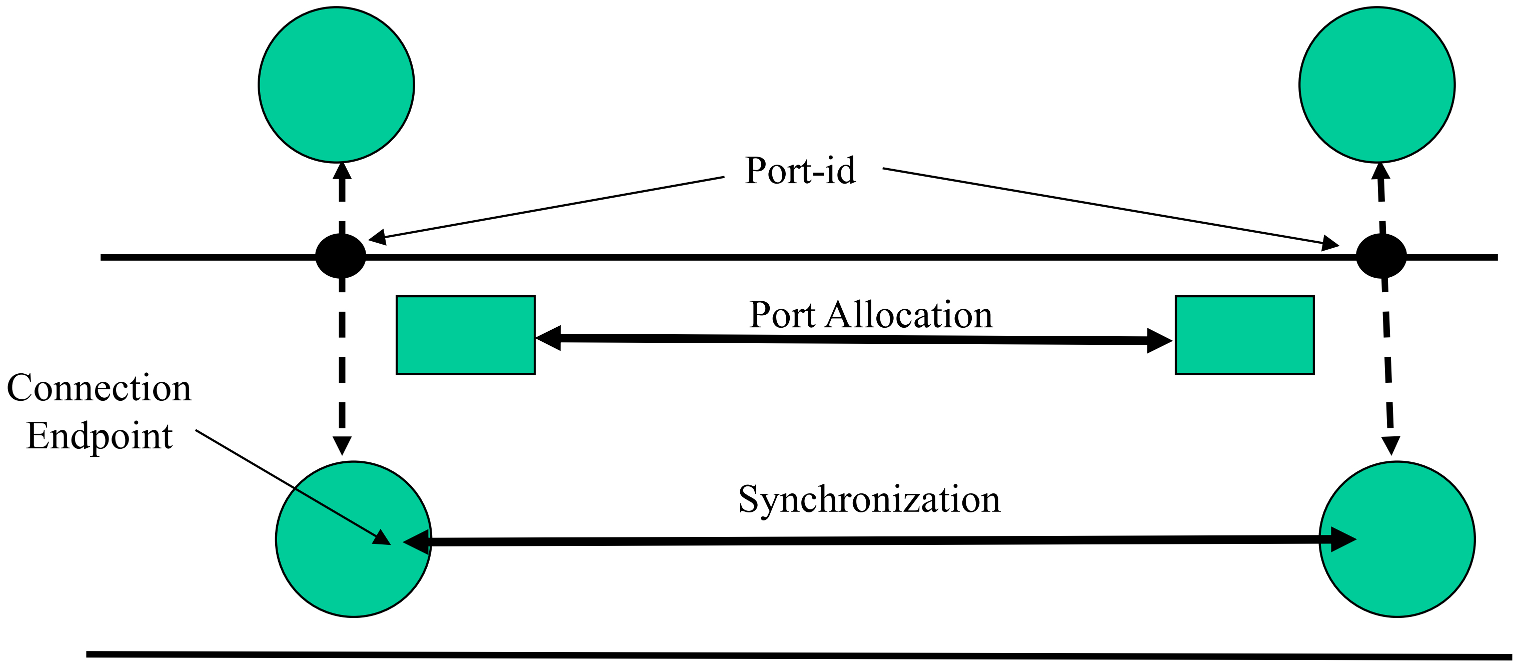
As we will see, TCP bounds MPL explicitly but assumes the other two are bounded, consequently isn't as robust.

The Theorem Exposes a Flaw



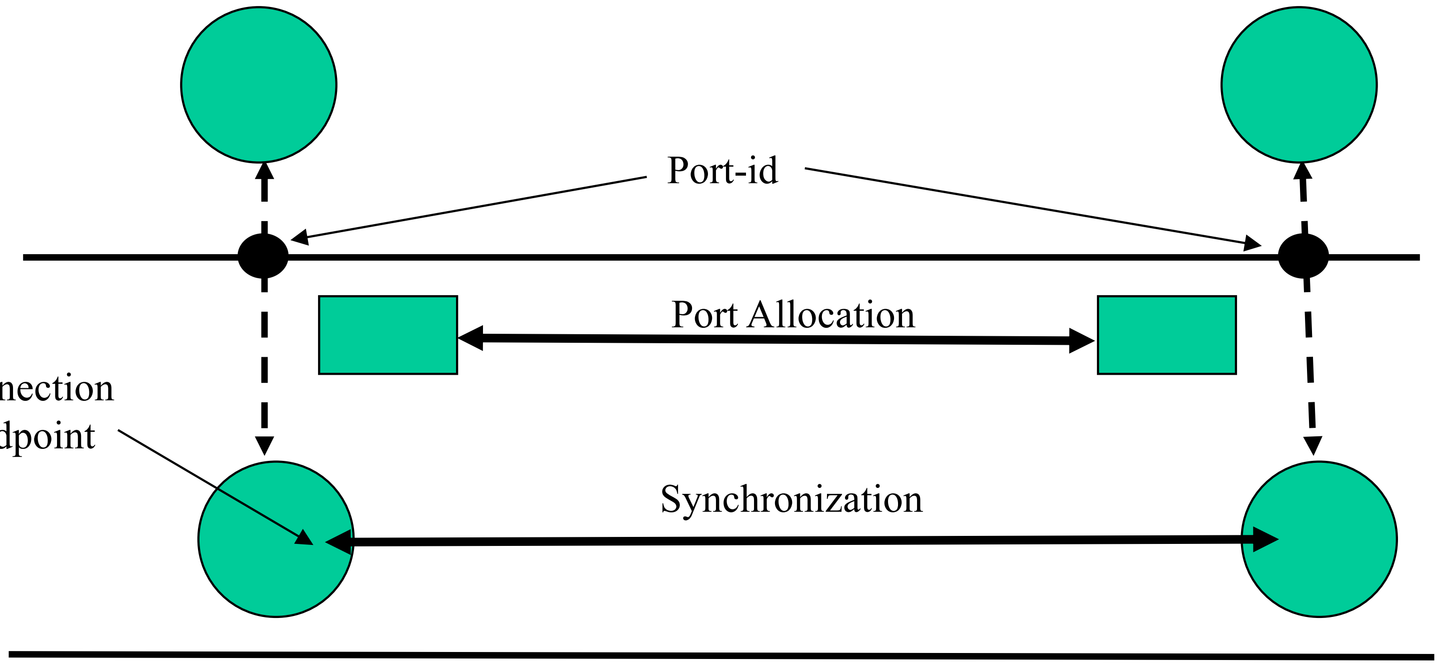
- In the Internet, a connection starts at sender's port and goes to the destination's port.
  - This is what connect/disconnect does.
- This combines port allocation and synchronization
- What Watson is saying when he assumes:
  - all connections exist all the time.
- Is that Port allocation and Synchronization are distinct.

This implies Separating the Two



- Instead, delta-t works differently:
  - Ports are allocated
  - Then protocol machines create synchronization (shared state)
  - And connection-end points are bound to the port-ids.
- Not conflating the two has significant implications.
  - No traffic for 2Δt, connection disappears, but ports remain allocated
  - Bindings are local; many security attacks found in TCP no longer work
  - Multiple connections can be bound to the same ports; providing security is simpler.

Port Allocation and Synchronization



- Port Allocation is a Management Exchange to Allocate Ports at Each End.
- Since Synchronization is based only on bounding 3 timers, once ports are allocated simply start sending.
  – The first PDU in each new sequence sets a flag that delta-t calls the Data Run Flag.
- If no traffic for 2Δt then discard the state.
  – The ports remain allocated even though there is no synchronization state in place.
- When all is completed, a management operation deletes the ports.
  – Synchronization state will disappear naturally after Δt

But Watson Implies an Even More Fundamental Result

- It defines the bounds of networking.
- It is IPC (networking), if MPL can be bounded.
- If MPL can not be bounded, it is remote storage.
  - it's a file transfer

Revisiting Retransmission Control

- The Semantics of Ack
  - Not going to be re-transmitted
- Selective Ack, ack some but not all.
  - Can reduce retransmissions with a large bandwidth/delay product.
  - Has implications for the value of A
- Bounding A, greatly limits Ack policy
  - Larger A increases RTT and MPL.
  - There are uses for large A, checkpoint recovery.

How Many Kinds of PDUs Do We Need?

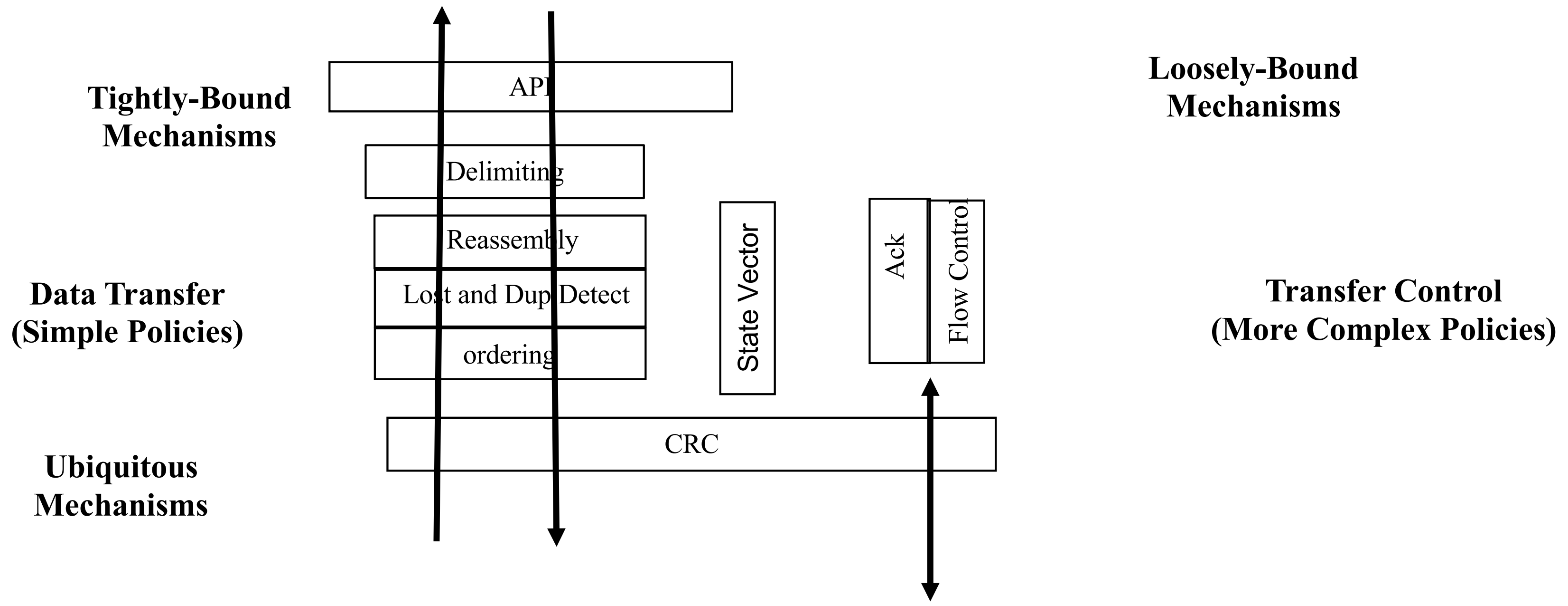
- This has been a controversial question.
- Clearly, we need at least one to carry the data.
  - A Transfer PDU is definitely required to carry the data.
- Is that all? Let’s look at our mechanisms.
  - Each one uses information in the PCI.
    - What information has to be with the data?
    - What doesn’t *have* to be with the data?
    - Is there any reason for it to not be?
- This turns out to be a very interesting distinction.

Three Kinds of Mechanisms

- Tightly Bound Mechanisms
  - Those that Must be with the Transfer PDU
    - Sequencing, Fragmentation, etc.
  - Policy is imposed by the *Sender*
- Loosely Bound Mechanisms
  - Those that don't Have to be with the Transfer PDU
    - Retransmission Control and Flow Control
  - Policy is imposed by the *Receiver*
- Ubiquitous Mechanisms (Tightly bound)
  - Those that have to be with all PDUs
    - Delimiting, CRCs

This Leads to

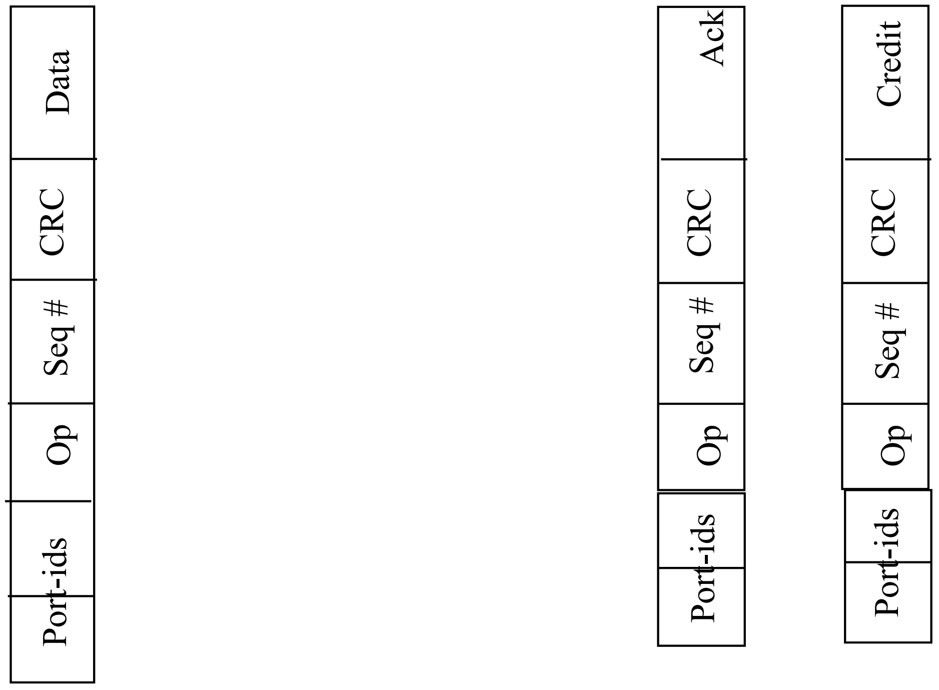
Protocol Structure (1)



- Using the distinction, it naturally cleaves into simple operations for data transfer and more complex ones for control implying separate PDUs for each.
- Control PDUs go no further, only Transfer PDUs “pass” to the layer above.
- Notice that “data” and “control” are only loosely coupled.
  - They are almost independent.
- In some cases, some loosely coupled mechanisms might not be present.
  - In which case, those PDU types are simply never generated.

So the Protocol Might Have

A Transfer PDU and at most one PDU for each loosely coupled mechanism.



Transfer or Data PDU

Control PDUs

Notice with this structure the only difference between a Transfer PDU for a reliable protocol and a connectionless unreliable protocol is whether Sequence Number is used for Ordering, i.e., whether the ordering policy is null or not and whether loosely bound mechanisms are used. If not used, they are simply never sent.

Syntax

(For some reason, everyone's favorite topic)

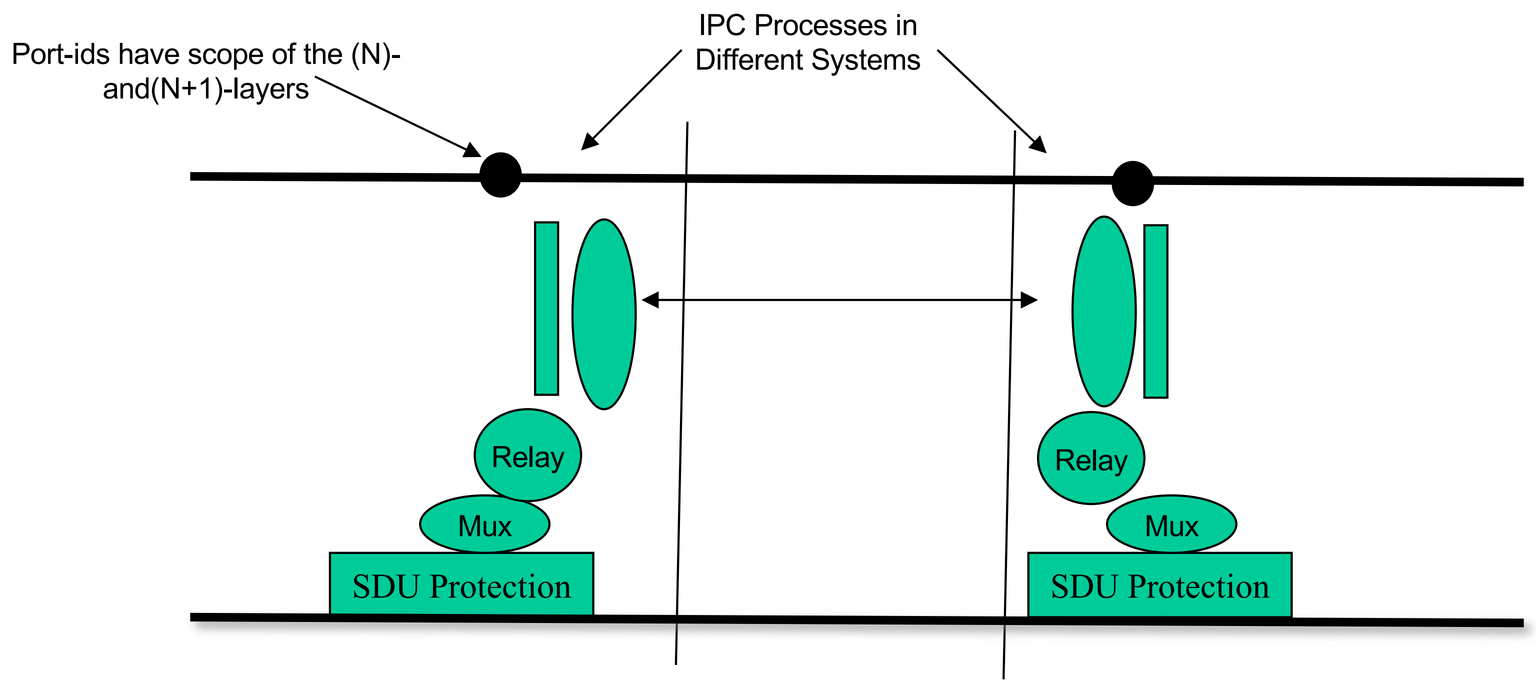
- Syntax of lower layers should avoid variable length fields
  - If required, put them at the end of the PCI
- Avoid complex or variable structures in IPC protocols
  - If necessary, use a type, length, value structure
- Alignment on byte boundaries is also important
  - Wasting a few bits can facilitate processing
- Abstract Syntax with different Encoding Rules
  - Many see it as merely another encoding scheme
  - Actually, a tool for making protocols invariant with respect to syntax
  - Current encoding rules are more efficient than hand-crafted ones.
- Different field lengths in data transfer protocols are not a big deal
  - Identifiers, used to index a table (look something up).
  - Integers, the only difference is the modulus of the arithmetic.
  - Bits, used as flags
- The *Procedures* are the Same.

Only One Data Transfer Protocol

- By separating mechanism and policy and
- By making the protocol invariant with respect to syntax, and
- decoupling data transfer and control thru the state vector

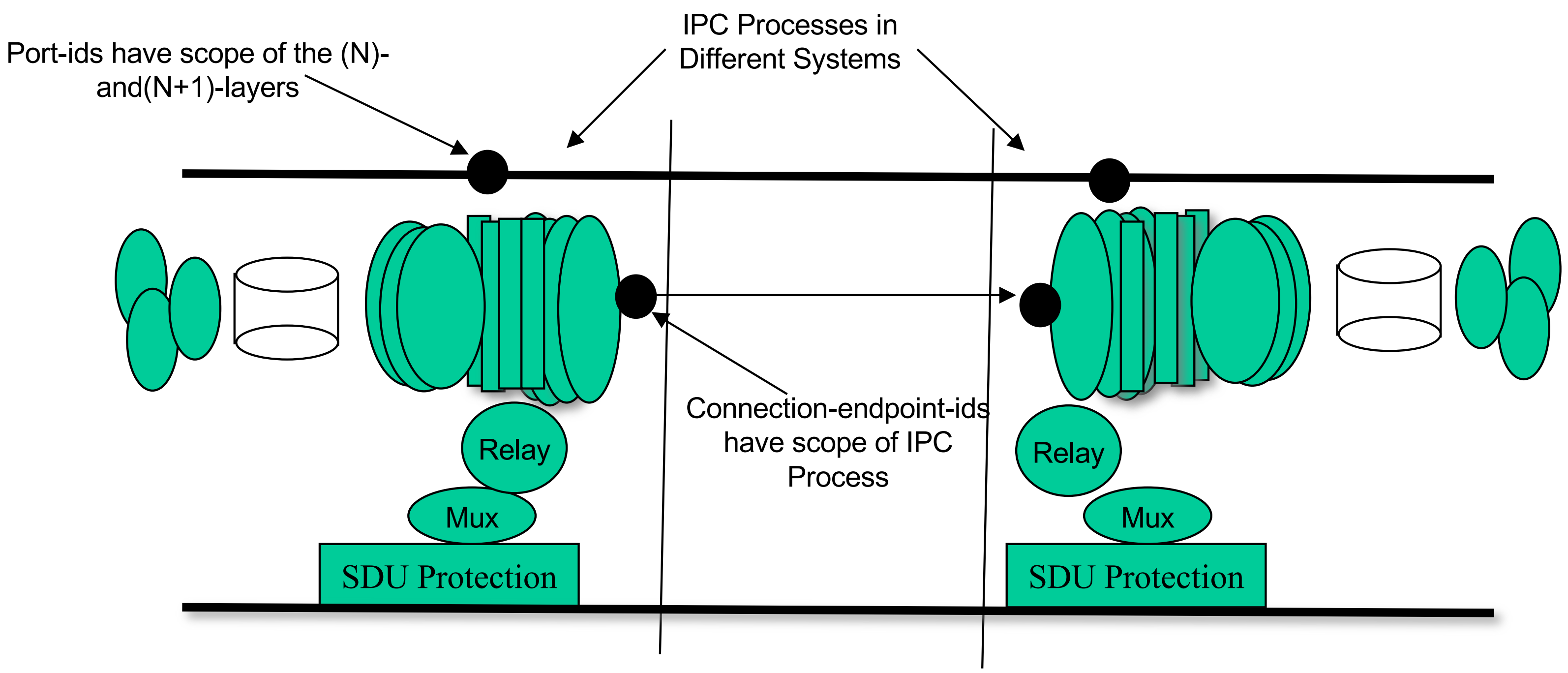
- We have made the argument that there is a simple implementation for a single data transfer protocol that can be configured to cover the entire range of protocols.

What We Know About Protocols and Layers: I


Port-ids have scope of the (N)- and(N+1)-layers
IPC Processes in Different Systems
Relay
Mux
SDU Protection
Relay
Mux
SDU Protection

- The necessary (and usually sufficient) condition for a layer is a Locus of Distributed Shared State With bounded MPL
- Port-ids are necessary and sufficient to ensure layer separation and independence.
- There may be multiple Layers of the same scope, but the functions assigned to the layers must be independent
- There is at most 1 SDU Protection and 1 multiplexing application per (N-1)-port-id. (property of the layer, not the protocol).
- There is one Relay per IPC Process. The relay must be able to see all (N-1)-ports.

What We Know About Protocols and Layers


Port-ids have scope of the (N)- and(N+1)-layers
IPC Processes in Different Systems
Connection-endpoint-ids have scope of IPC Process
Relay
Mux
SDU Protection
Relay
Mux
SDU Protection

- There is one Error and Flow Control state machine per flow allocated.
  - Feedback mechanisms require Data Transfer Control and bounding 3 timers.
  - For security requirements, all connections must have flow control.
- Watson implies decoupling ports and connection-endpoints.
- Data Transfer functions are decoupled from Data Transfer Control through a state vector
  - which are similarly decoupled from Layer Management (Resource or Management Information Base).
    - Not so much layering within the same scope as orthogonal separation into 3 loci of processing.
- If any of these rules are broken, there will be problems with the design.

Protocol Specification

- A Protocol Specification is implementation-independent.
  - Informal and Formal Methods
  - Both are necessary
- A Specification is a Reference, not an article about the protocol
  - Informal Method - The Outline Form
  - Formal Methods - Three Basic Models
    - Finite State Machine - see Estelle, SDL, etc.
      - Pure Petri Nets are for verification, but not complete for specification
    - Temporal Ordering - see LOTOS, etc.
    - Language Models - see pi
- FDTs prove progress, absence of deadlocks, races, automatic code generation, etc.
- A Formal Method that is more complex than the code has a higher probability of bugs!
- A Formal Specification can not be considered definitive, merely less ambiguous.

Design Issues (1)

- Determine the range of operation of the protocol
  - Near the media, its properties will dominate.
  - Near the application, its requirements dominate.
- Determine the service expected from the layer below.
- Primarily QoS issues.
- How much shared state is required?
  - Normally if loosely bound functions are present
- Stream or SDU
  - Usually SDU, even for “streaming video or voice”
- Maximum PDU Size?
- Number of PDU types?
- Opcodes (Type Field) vs Control Bits
  - Usually opcodes; too many illegal combinations with control bits

Design Issues (2)

- Sequencing
  - Size of Sequence number space and Credit Field
  - Bits, byte or packet sequencing
  - Initial Sequence number
- Implications of size of the Sequence Number and Credit
  - Want to keep the pipe full but avoid two sequence numbers in the net at the same time taking into account retransmissions.
  - Packets/second * Round Trip Time ((data rate)/8*MTU)*RTT $\ll \Delta t$
- Retransmission strategy
  - Ack and maybe Nack
  - Piggybacking, not likely
- Establishing Time-out ranges
- Flow Control strategy
  - About 1980 flow control shifted from the discrete to the continuous.
    - from counting buffers to process control

Implementation Issues

- Minimize context switches
- Minimize data copies
- Is the protocol a process, a thread, or a library.
- How to wait. Buzzing is not a good idea. Semaphores or monitors are.
- Interface flow control
- Buffer management

Pooled vs Static Buffers

- From Denning, Peter. "A Statistical Model for Console Behavior in Multiuser Computers" CACM, Vol.11, No.9 p.605, Sept 1968.
- For 50 users and a ratio of characters/interrupt = 10.

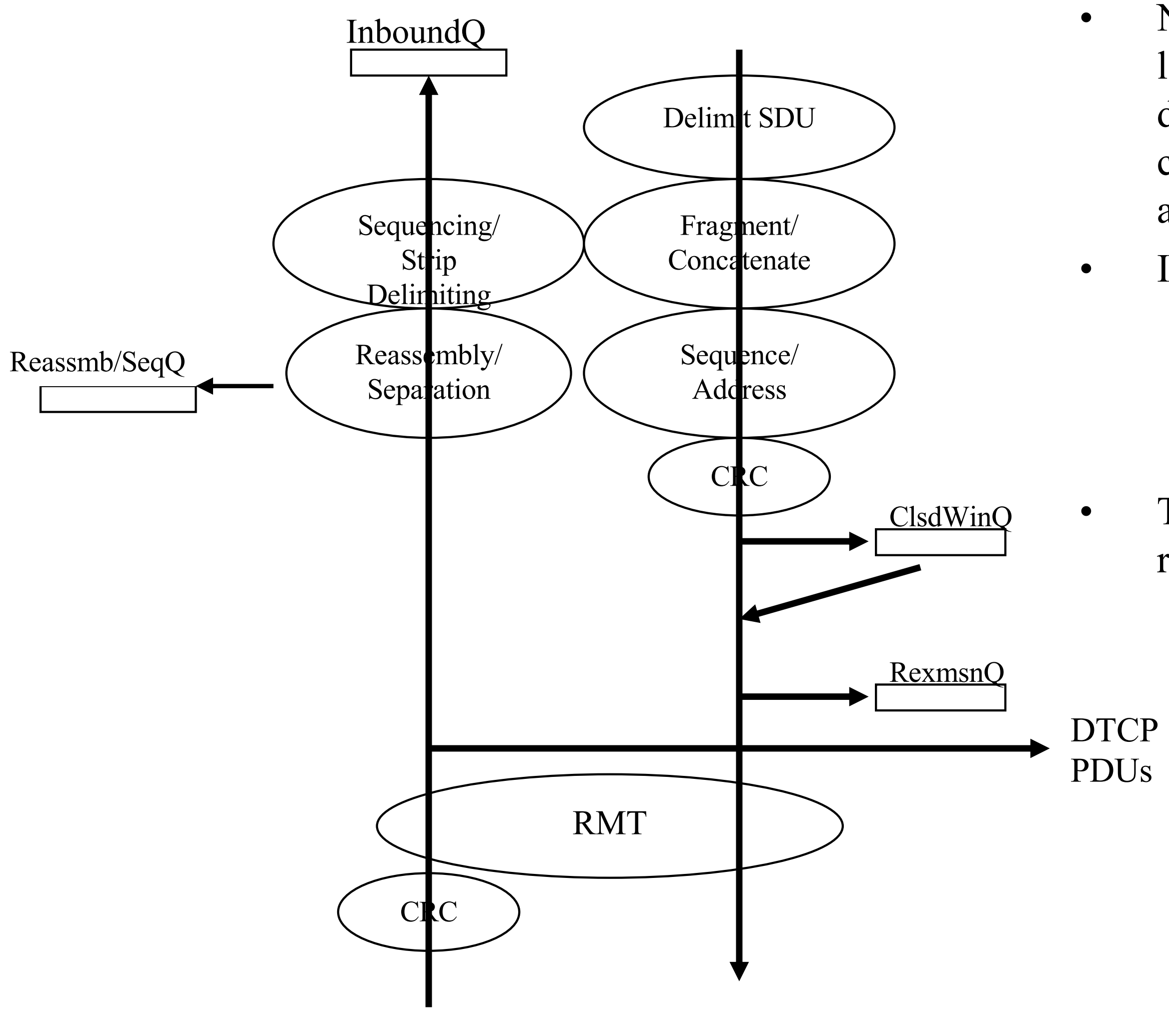
Total Buffer Size (in bytes)	**Probability of Overflow** **Pooled**	**Static**
750	.006	.90
1000	10^{-6}	.76
1500		.44
2000		.22
2500		.10
3000		.05
3500		.02
4000		.01
4500		.004

Blank because the numbers were too small to be represented by the computer.

It takes to here with static to get to where we started with pooled.

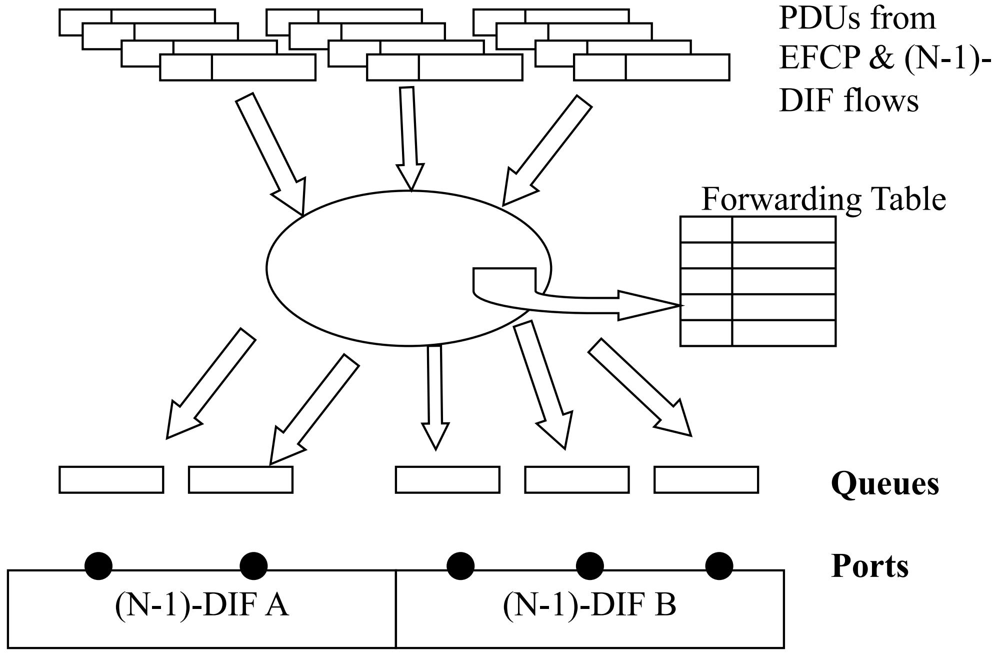
- This result is completely general for static vs pooled resources. It is really a no-brainer: Always pool buffers, unless there is a good reason to ensure it will run out.
  - Takes at least 6 times more memory for static vs pooled and it isn't linear.
  - At today's requirements the difference will be measured in orders of magnitude.

Data Transfer Protocol



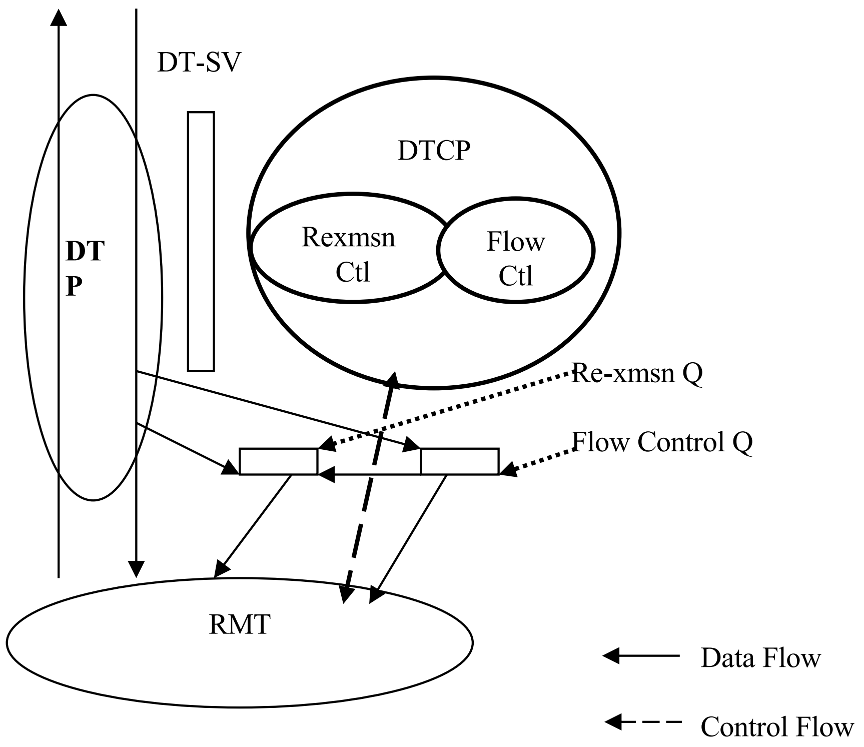
- Notice that the flow is a straight shot, very little processing and if there is anything to do, it is moved to the side. The most complex thing DTP does is reassembly and ordering.
- If there is a DTCP for this flow:
  - If the flow control window closes, PDUs are shunted to the flow controlQ.
  - If the flow does retransmission, a copy of the PDU is put on the rexmsnQ.
- These PDUs are now DTCP's responsibility to send when appropriate.

Next the Relaying and Multiplexing Task



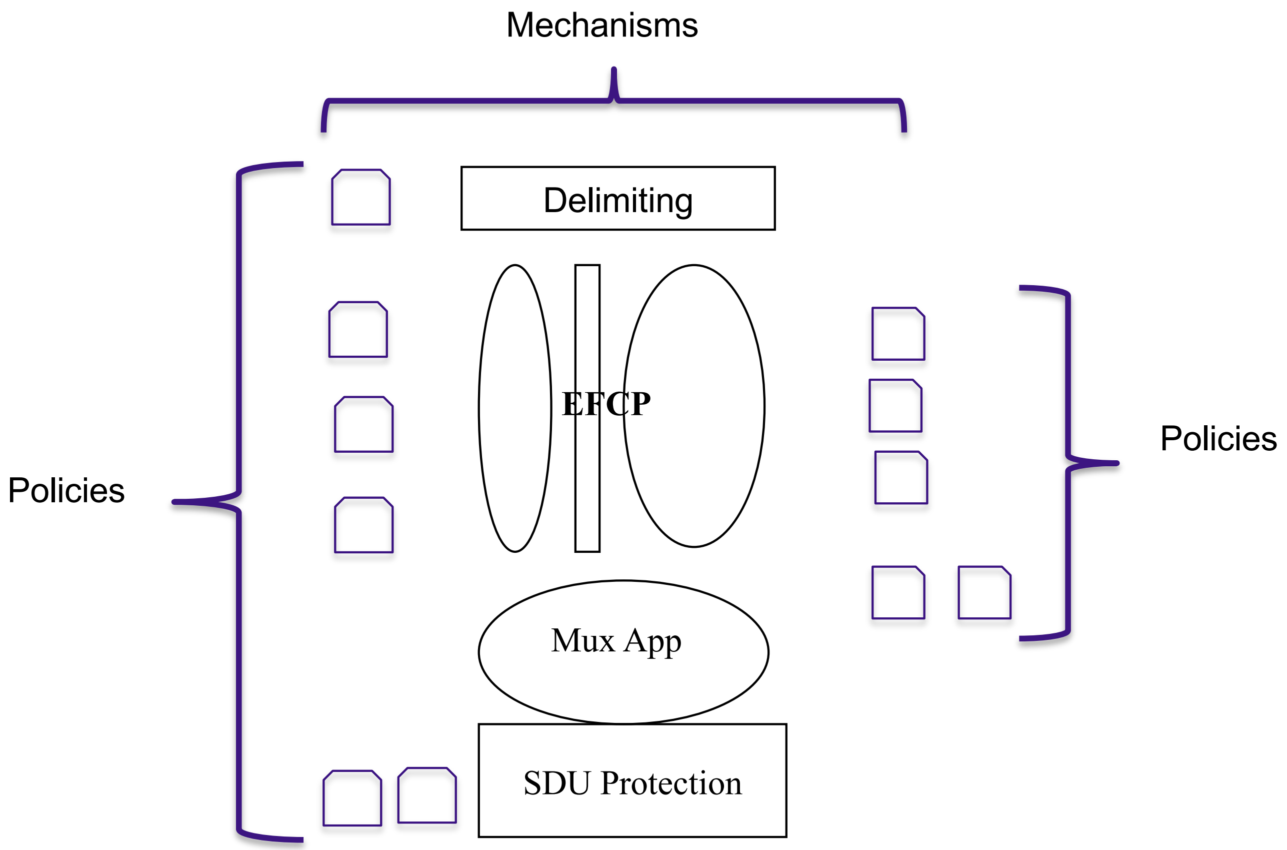
- Outbound this is the first queuing we must hit and even here it may not be necessary
- DTP flows are classed by the connection-id, RMT policy determines the servicing of the queues, for each PDU consulting the forwarding table and posting it to the proper (N-1)-port.

Finally Data Transfer Control Protocol



- For flows with retransmission (acks) and/or flow control, a DTCP is required.
- DTCP controls flow volume, the RMT controls flow rate of (N-1)-flows.
- We need to bound 3 timers, MPL, Ack delay, and retries.
- For the rest of the course, we will be looking at how current protocols follow this model.

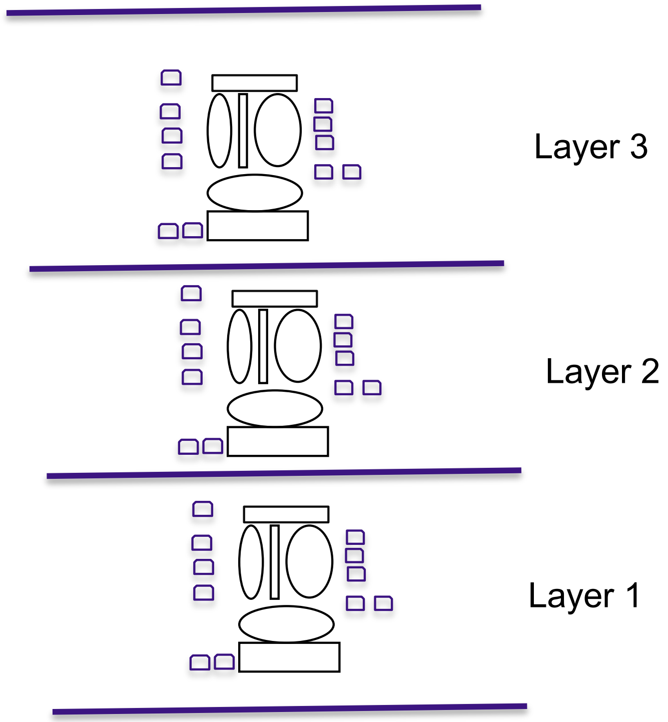
The Structure of Data Transfer Protocols



We have found that this structure repeats
in Layers of Different Scope,
But what is the way to implement it in a system.

Repeating Layers would seem to Imply

A pipeline like this would be the way to go



No, we can do much better

Consider the following

- Remember Automata Theory?
- You’re kidding, right?
- Nope!

- A **Turing machine (TM)** is a mathematical model of computation that defines an abstract machine[1] that *implements a specific algorithm* which is executed by manipulates symbols on a semi-infinite tape.
- A **Universal Turing machine (UTM)** is a *Turing machine* that simulates an arbitrary *Turing machine* on arbitrary input. The *universal machine* essentially achieves this by reading both the description of the *machine* to be simulated as well as the input to that *machine* from its own tape.

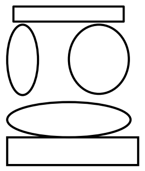
- Now Suppose:

Automata Theory Can Be Useful

- Now Suppose that:

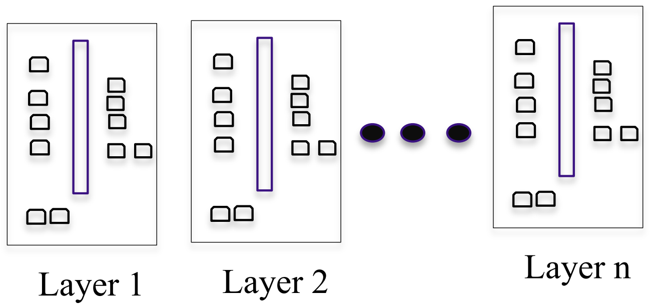
Isn't it interesting what comes from theory? Note that all of the insights have come from theory, not implementation.

The mechanisms are a UTM



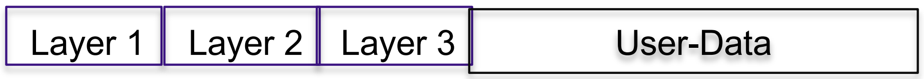
This is Silicon

The policies define a TM and there is one for each layer.



These are plug-ins of code and data

And of course, the PDU is the tape



Point at the policies for Layer 1, process the Layer 1 PCI, Move the tape down, Point at the policies for Layer 2, Process the PCI for Layer 2 and so on. No context switches, no data copies. This makes for a much more efficient, simpler implementation and a huge collapse in complexity.

Is this possible?

Of course, Been There Done That

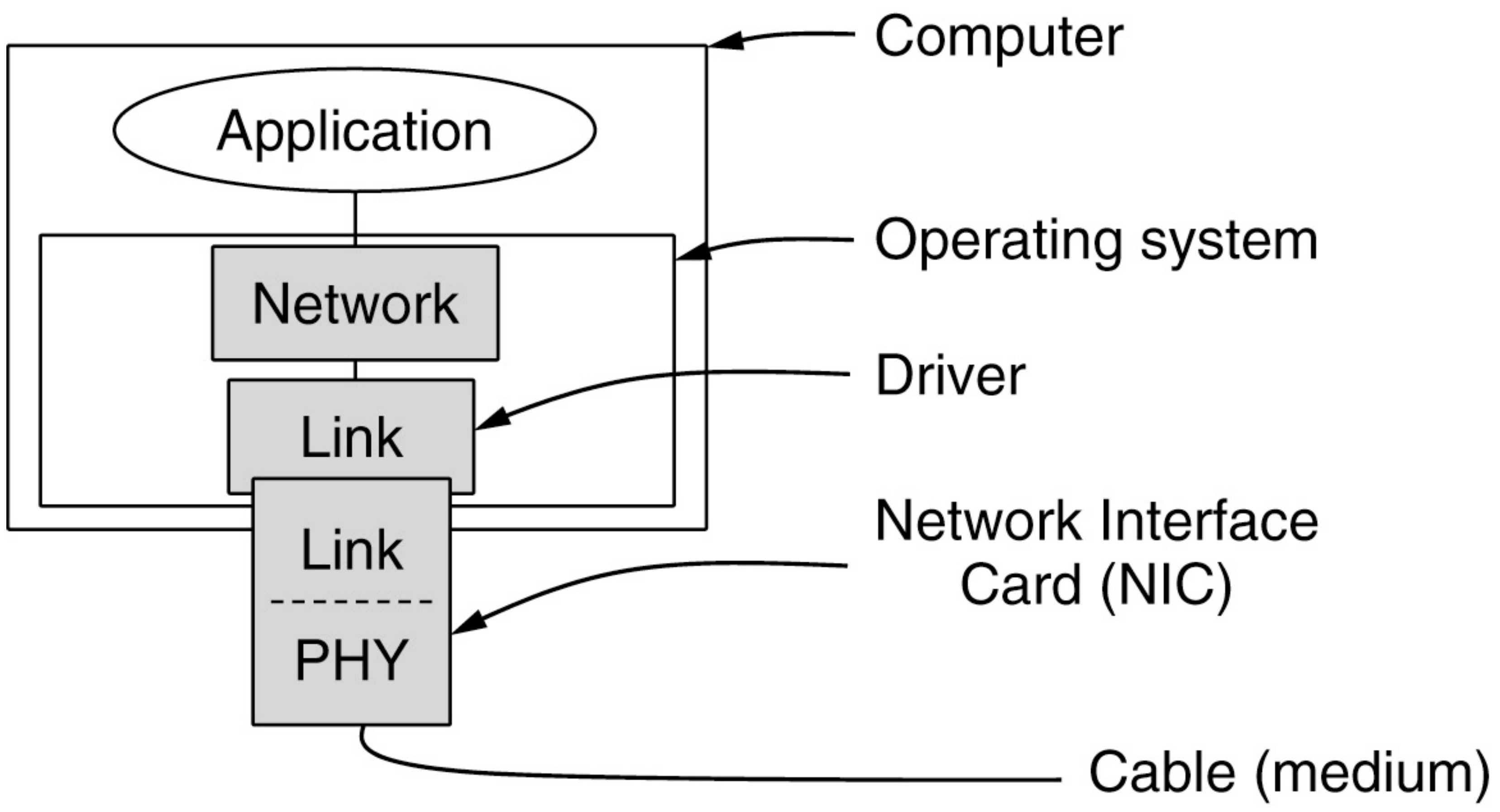
Elementary Data Link Protocols

- Assumptions underly the communication model
- Three simplex link-layer protocols
  - Utopia: No Flow Control or Error Correction
  - Adding Flow Control: Stop-and-Wait
  - Adding Error Correction: Sequence Numbers and ARQ

Initial Simplifying Assumptions (1 of 2)

- Independent processes
  - Physical, data link, and network layers are independent
  - Communicate by passing messages back and forth
- Unidirectional communication
  - Machines use a reliable, connection-oriented service
- Reliable machines and processes
  - Machines do not crash
- Unstated assumption: The upper layer will take any data as soon as it arrives.
  - This avoids needing two more commands and additional states.
    - Receiver Not Ready (RNR) and Receiver Ready (RR)

Initial Simplifying Assumptions (2 of 2)



Implementation of the physical, data link, and network layers

T’s Protocols: I

- Protocol 1: Utopian Simplex Protocol
  - What mechanisms? What policies?
    - 1 PDU type: Transfer, Send Policy: If data from user, send it.
- Protocol 2: Simplex Stop and Wait Protocol
  - What mechanisms? What policies?
    - 2 PDU types: Transfer, Ack. Send policy: If data from User and Wait for Ack. Ack Policy: As soon as user takes the packet ack.
- Protocol 3: Simplex Protocol for a Noisy Channel
  - What mechanisms? What policies?
    - 2 PDU Types: Transfer, Ack. Data Corruption: Checksum; Lost and Duplicate Detection (sequence number); Sender must wait for a positive ack. Timeout on no Ack after RTT+ε. As soon as PDU received send Ack.
  - But this has a rather major bug. What is it?
- Piggybacking isn’t all it is cracked up to be.

Applying Our Legos to T's Protocols

- First, Re-cap our Building Blocks
  - Delimiting
  - Packing an SDU into a PDU
  - Ordering/Sequencing
  - Retransmission Control (Ack)
  - Data Corruption
  - Lost and Duplicate Detection
  - Flow Control
- And the design issues we just covered (for our purposes we can ignore some).
- T describes 6 protocols,
  - but we will need to use our imagination

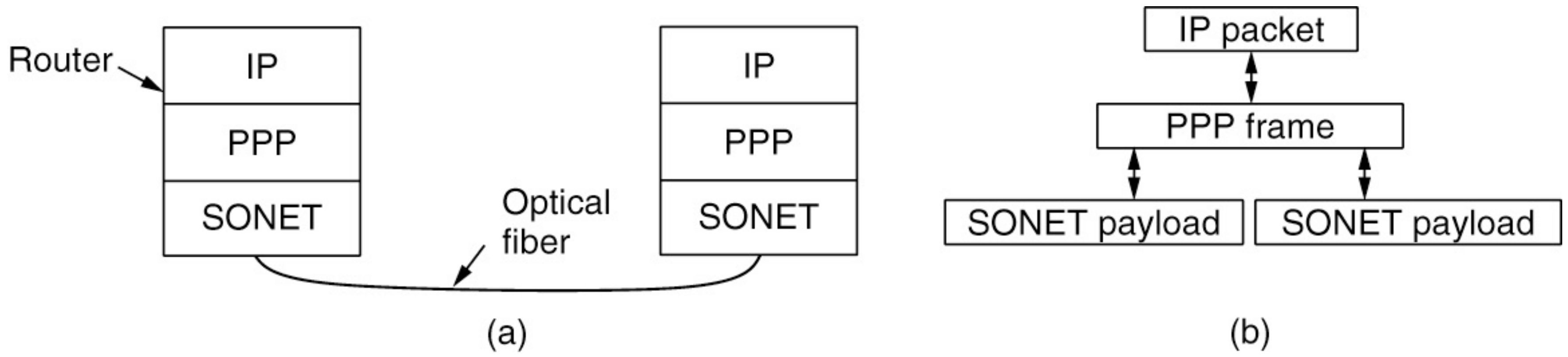
T’s Protocols: II

- Protocol 4: One-bit Sliding Window Protocol
  - What mechanisms? What policies?
- Protocol 5: Go Back N Protocol
  - What mechanisms? What policies?
- Protocol 6: Selective Repeat Protocol
  - What mechanisms? What policies?

Data Link Protocols in Practice

- Packet over SONET
- ADSL (Asymmetric Digital Subscriber Loop)

Packet over SONET (1 of 4)

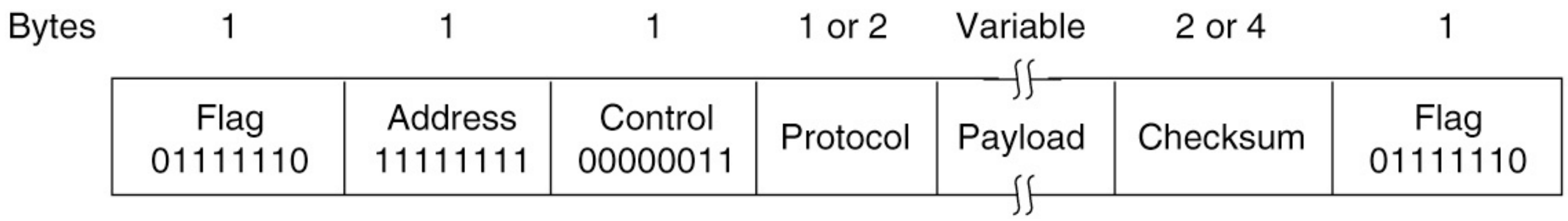


Packet over SONET. (a) A protocol stack. (b) Frame relationships.

Packet over SONET (2 of 4)

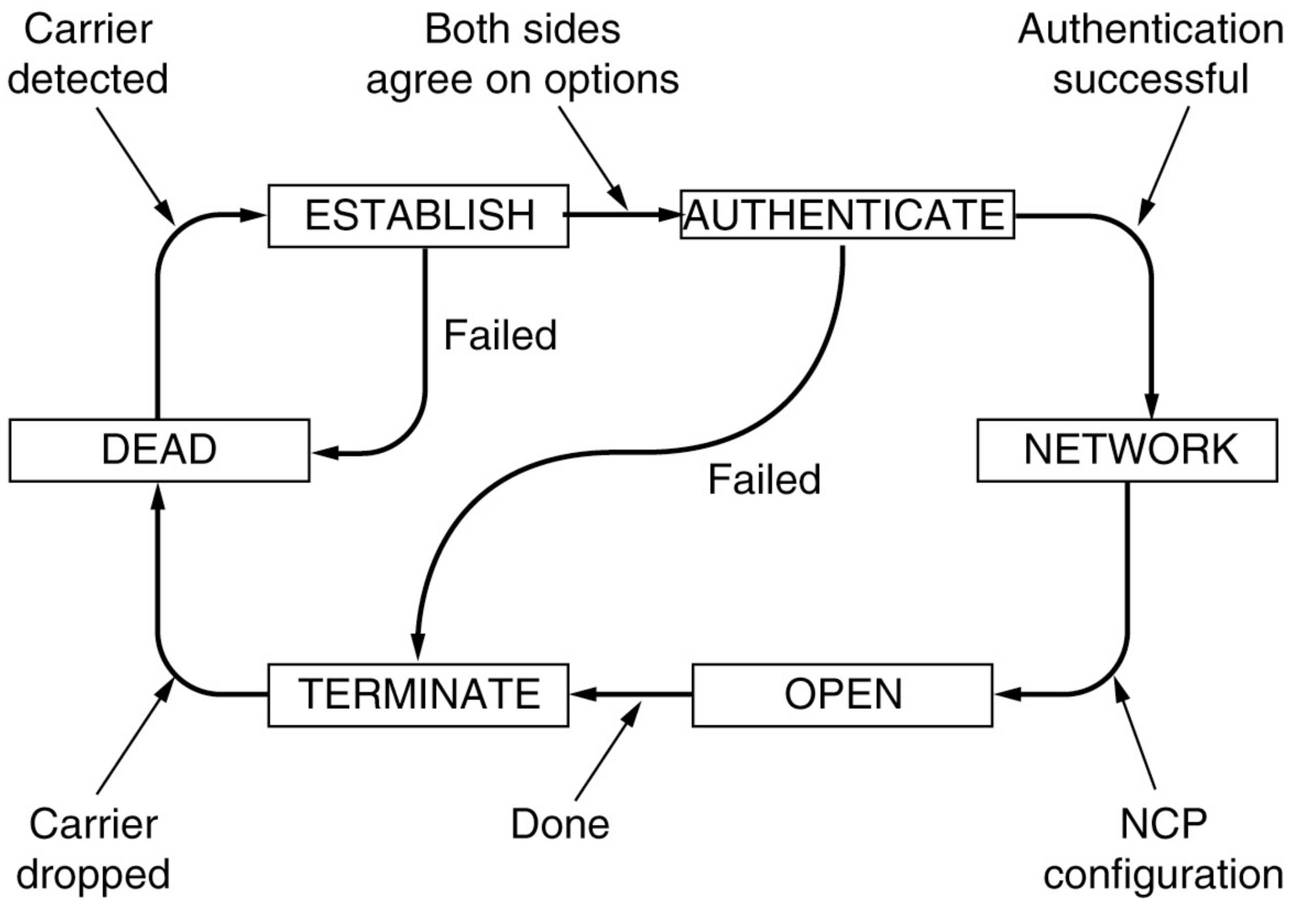
- PPP (Point-to-Point Protocol) features
  - Separate packets, error detection
  - Link Control Protocol (LCP)
  - Network Control Protocol (NCP)

Packet over SONET (3 of 4)



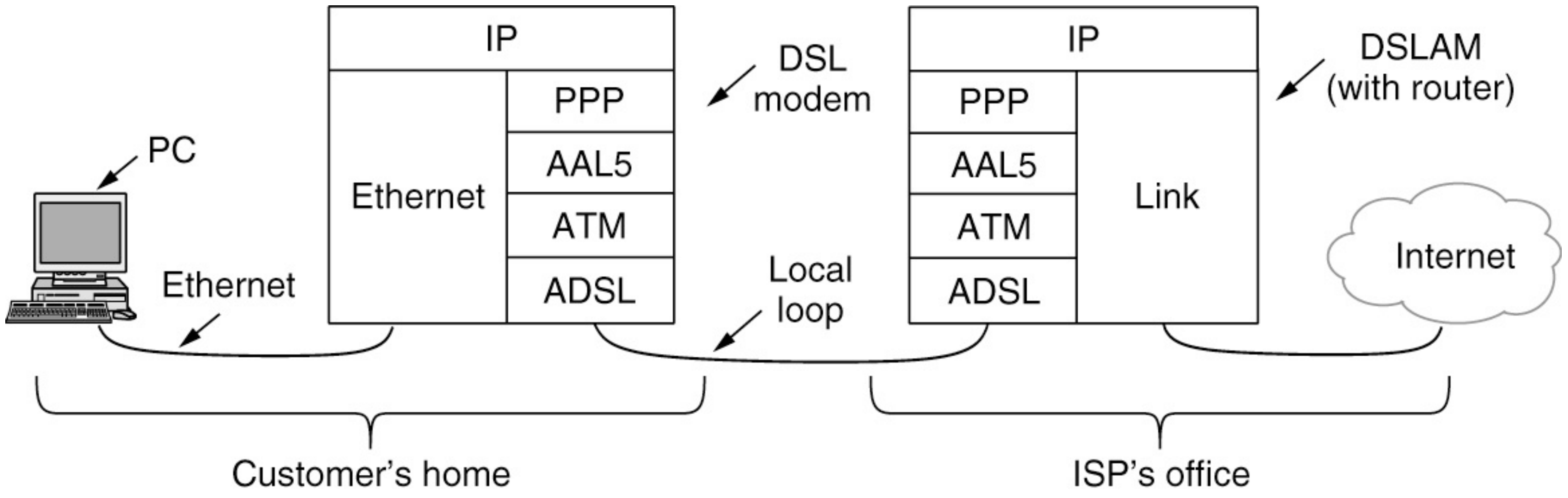
The PPP full frame format for unnumbered mode operation

Packet over SONET (4 of 4)



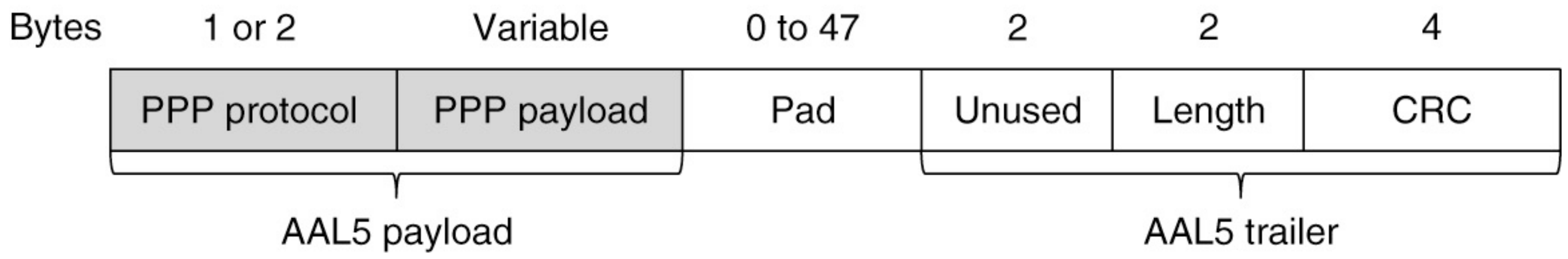
State diagram for bringing a PPP link up and down

ADSL (1 of 2)



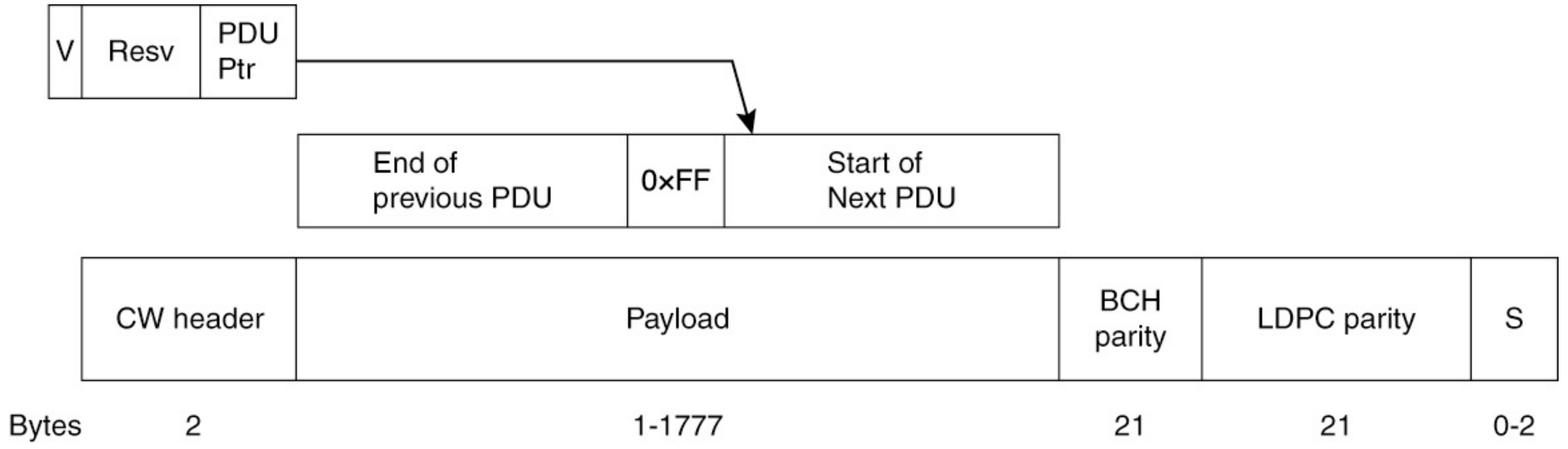
ADSL protocol stacks

ADSL (2 of 2)



AAL5 frame carrying PPP data

Data Over Cable Service Interface Specification (DOCSIS)



DOCSIS Frame to codeword mapping

Copyright