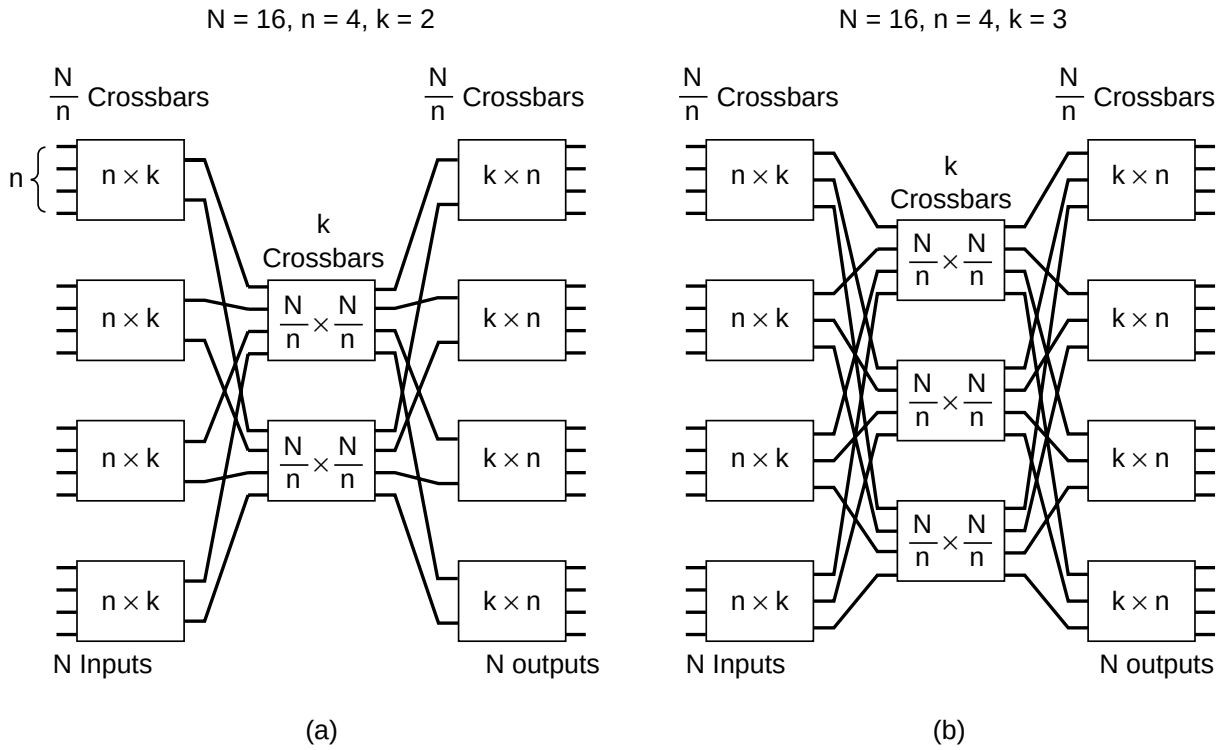
Announcements

- project description was handed out
- project #1 grades will be sent by email today
  - common errors:
    - forget to include some .h files
    - missing/wrong makefiles
  - re-grades
    - see the TA with a **working** version of your program

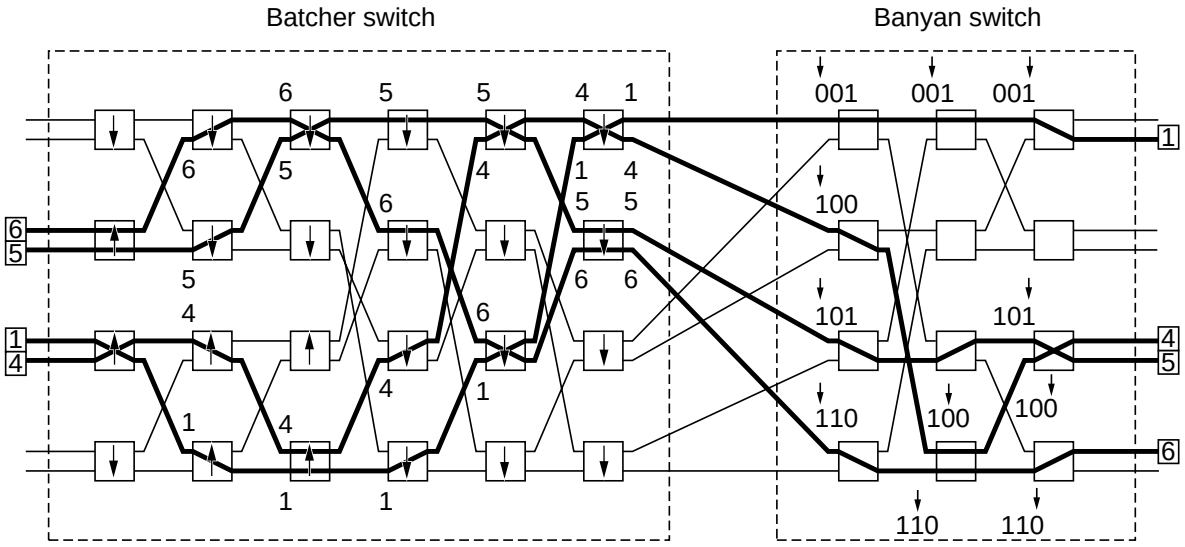
Switching Fabric (space division)

- Cross bars are great, but require $O(n^2)$ wires
- Can use a collection of smaller cross bar switches
  - penalty: a request to connect may **block**



Batcher-banyan Switching

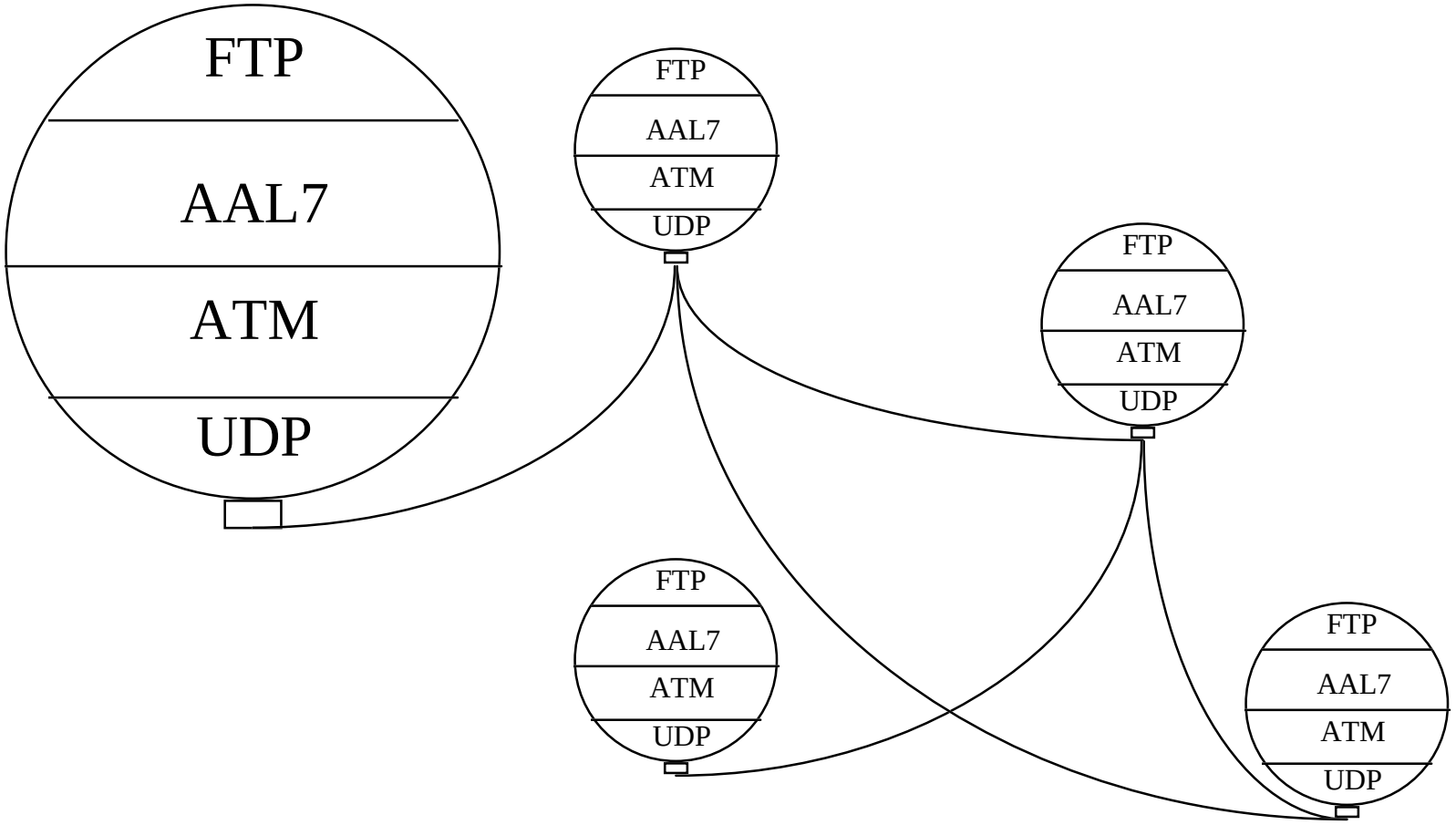
- Banyan
  - can do a “good” or “poor” job of switching due to collisions
  - if the inputs are sorted, we get performance
- Batcher
  - sorts traffic base on full address of destination
  - compares two colliding packets and uses final destination to select output port
  - requires $O(n\log^2 n)$ nodes (2x2 switching elements)



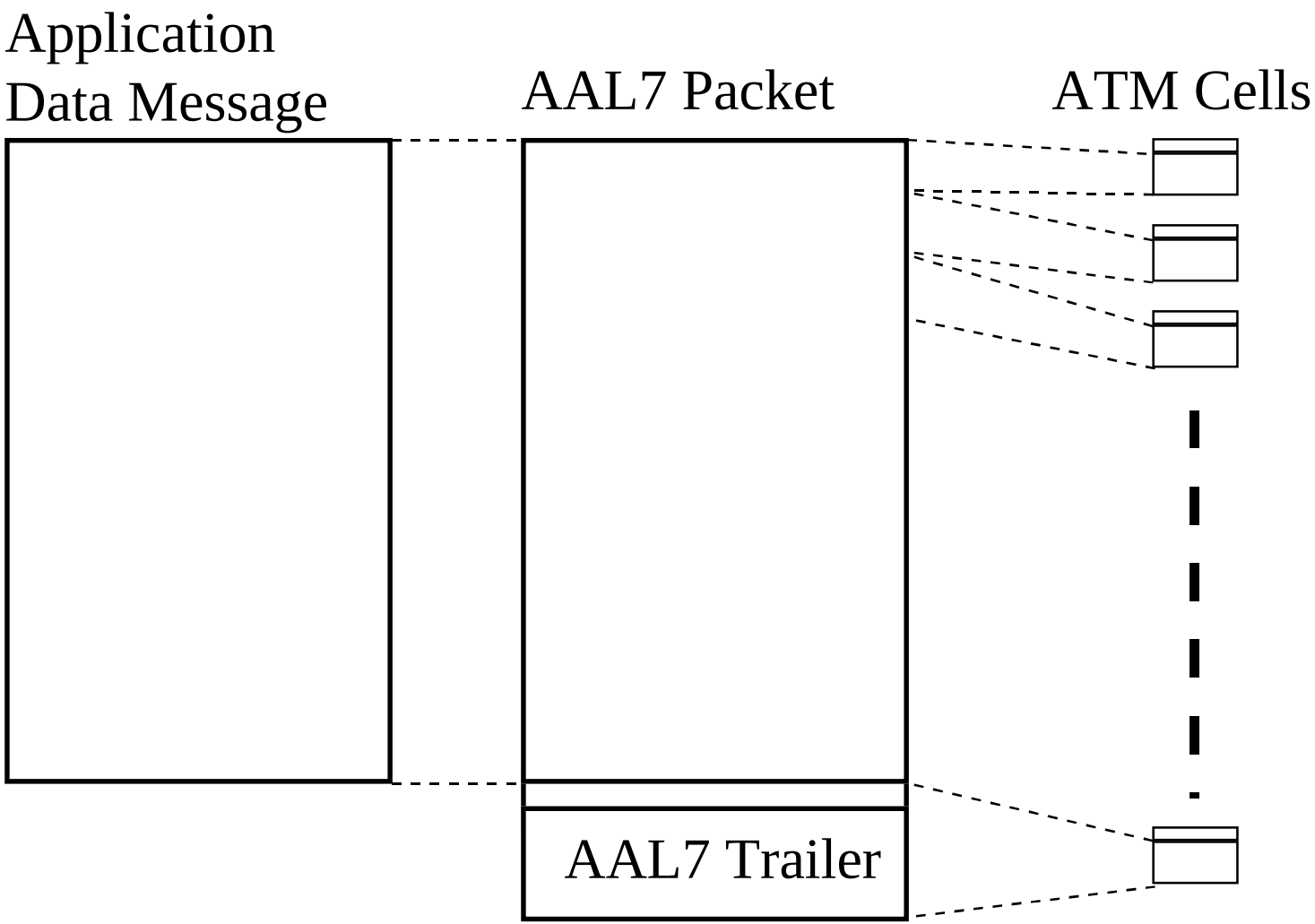
Project Introduction: Implementation of ATM Network Layer and Reliable ATM Adaptation Layer

based on a project developed by
Dr. Larry Landweber, University of Wisconsin

The Big Picture - System Structure



The Big Picture - Flow of Data



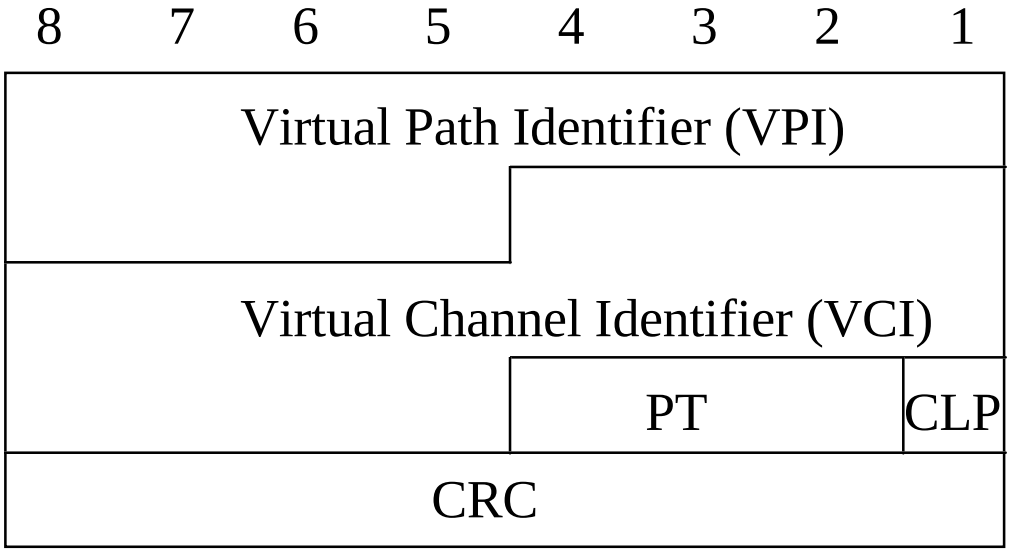
Some Terminology

- *connection-oriented* -vs- *connection-less*
  - Does it look like there is a wire?
- *reliable* -vs- *unreliable*
  - Is data guaranteed to get there?
- *service interface*
  - What a layer offers to its users.

ATM Layer

- Provides connection-oriented, unreliable service interface
- Uses a virtual circuit mechanism
- Requires a signalling protocol
- Also needs a routing protocol

ATM Cell Format



- ATM cells are 53 bytes
  - 5 bytes of header
  - 48 bytes for payload

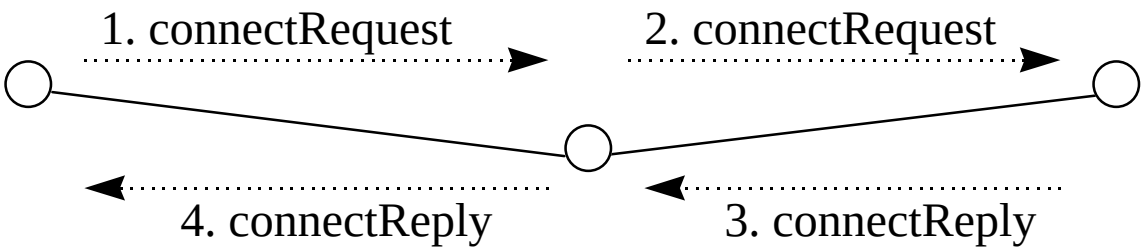
- VPI, CLP, and CRC fields can be ignore for this project

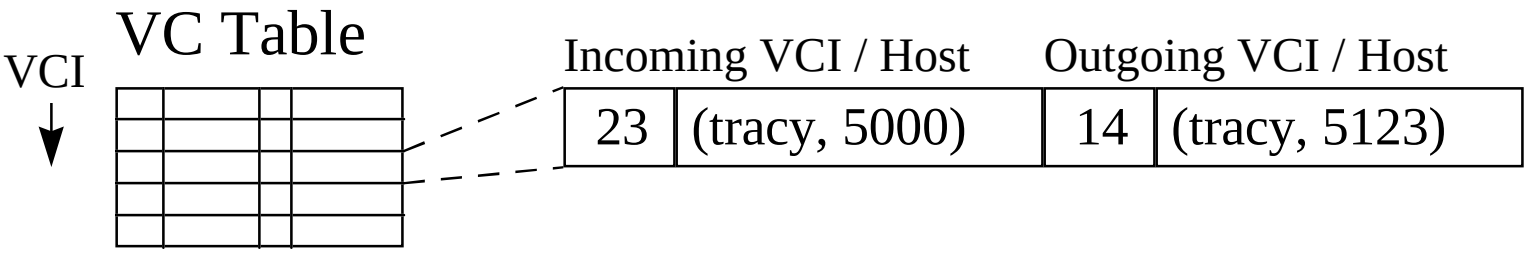
- Possible uses of PT
  - signalling cell
  - last cell in packet

Signalling Protocol

- Establishes full-duplex ATM virtual circuits (VC's)
  - Also used to tear them down
- Must indicate that an ATM cell is a signalling cell
  - permanent virtual circuits
  - special payload-type (PT) value
- Must access network routing tables
- Signalling mus be reliable
  - even though UDP is not
  - must do retransmission when cells are lost

ATM Signalling Example





ATM Signalling Example (cont.)

Step 1 Source sends connectRequest message with
- outgoing VCI for reverse channel
- destination node's ID

Step 2 Intermediate switch receives connectRequest message
- checks routing table to find outgoing host in forward direction
- allocates a VC table entry
- sends connectRequest message to next switch

Step 3 Destination switch responds with connectReply
- includes VCI chosen for forward channel

Step 4 Intermediate switch receives connectReply
- if connection allowed, fixes VCT and forwards another connectReply message back to the source

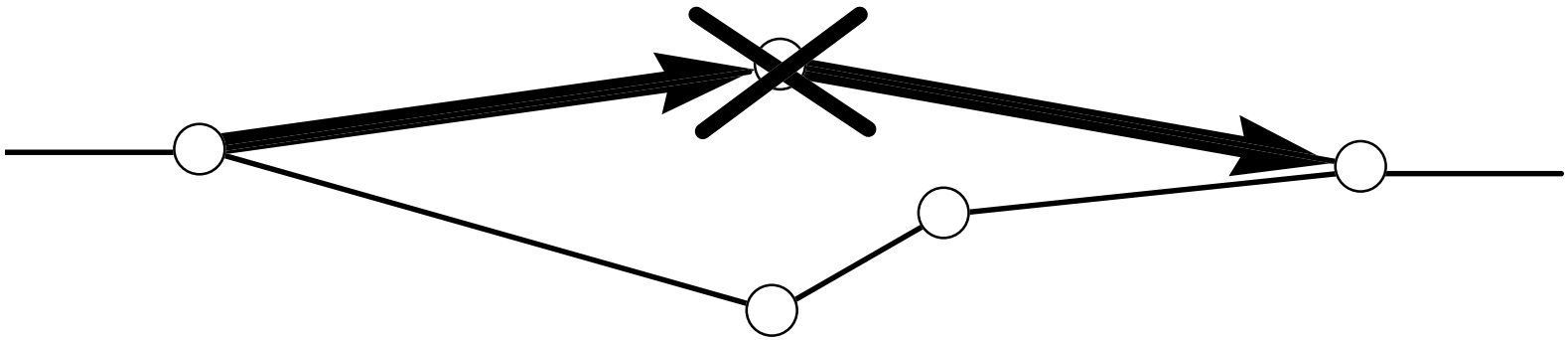
Signalling Design Requirements

- What are your message formats?
  - connection request / reply
  - disconnection request / reply
- What is protocol for:
  - connection establishment?
  - connection teardown?
- How to handle lost signalling cells?
- How to identify duplicate
  - connection establishment
  - connection teardown requests?

ATM Routing

- Signalling code needs to
  - know where to forward cells for a given destination
  - keep next hop information in a routing table
    - separate table for each node
    - each table tells next hop for every other node
- Routing Table
  - stores info about how to get to different destinations
  - need a routing protocol to build and maintain routing tables
- Routing Protocol
  - **Link-state** - each node periodically floods local link costs to all other nodes, then runs a shortest-path algorithm
  - **Distance-vector** - each node sends its neighbors reachability and distance info about all other nodes; if a node learns of a shorter path, it updates its distance matrix
  - OR roll your own...

Adaptive Routing



- After a node failure:
  - Must fix routing tables
  - Sometimes must also patch VC's without dropping the connection

Routing Protocol Design Issues

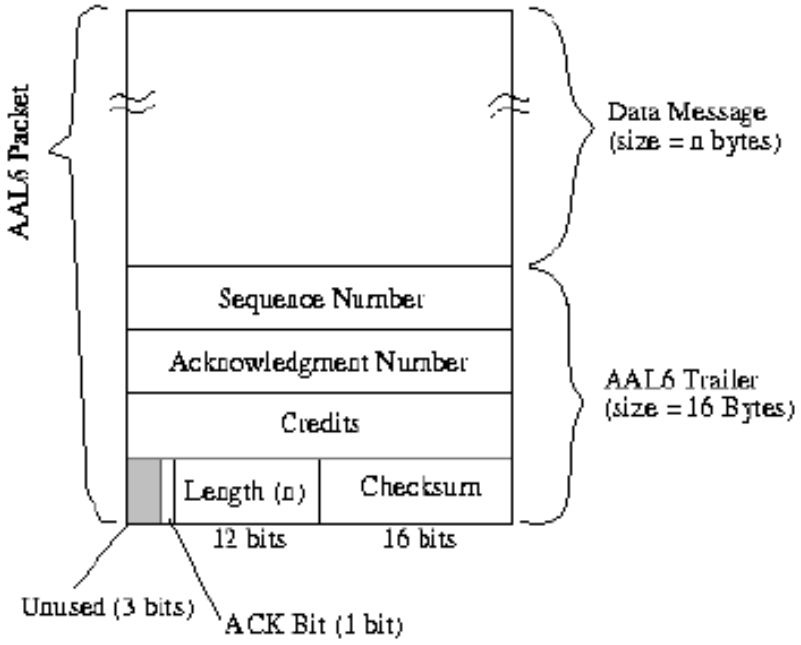
- Must have a mechanism to detect link status
  - periodically send “ping” packets
  - link metric can be 1 or ∞
- Link-state design problem:
  - How to limit extent of link-state message flooding?
- Distance-vector design problem:
  - How to stabilize routing tables quickly?

AAL7 Layer

- Provides
  - connection-oriented
  - reliable byte-stream service to application layer
- Uses
  - connection-oriented
  - unreliable service provided by ATM layer
- Like TCP

AAL7 Packet Format

- Each message from application is encapsulated in an AAL7 packet
- A trailer is appended for:
  - Flow-control
  - Acknowledgment
  - Error detection



AAL7 Flow Control

- Goal: don’t swamp the receiver
- Receiver needs to advertise its window size (credits)
- Sender should adjust its window based on receiver’s advertised window size
- Use sliding window protocol (like TCP)

AAL7 Segmentation and Reassembly

- Segmentation - outgoing
  - packets must be split into ATM cells
- Reassembly - incoming
  - ATM cells must be assembled into complete AAL7 packets
  - inspect VCI field and assemble into appropriate packet buffer
- Challenge: *Can you minimize or eliminate copies?*

AAL7 Checksum Computation

- Could use TCP's algorithm
  - form 1's complement addition over 16-bit units of the message (including the trailer)
  - checksum is 1's complement of above computation
- ... OR roll your own

AAL7 Reliability Issues

- Acknowledgments
  - Cumulative or selective?
- Sequence numbers
  - Counting what?
  - How to handle wrap around?
- Retransmission of AAL7 packets
- When to drop a connection?

AAL7 Service Interface

`int aal7_connect()`

- An active request to establish a connection to a remote Service Access Point (SAP)
- Returns a descriptor to be used in future calls to represent an endpoint of communication
- Blocks the caller

`int aal7_disconnect()`

- Does what you would expect

AAL7 Service Interface

`int aal7_listen()`

- Used by servers to register a service
- Returns a descriptor used as the argument to `aal7_accept()` call
- Note: this descriptor is different (it's a SAP descriptor)

AAL7 Service Interface

`int aal7_accept()`

- Indicates that a server is willing to accept a connection from a client
- Blocks until a connection is established
- Returns a descriptor that can be used to communicate with the client

AAL7 Service Interface

`int aal7_send()and aal7_recv()`

- Both require a valid connection descriptor and a pointer into a buffer
- Both block the caller

AAL7 Service Interface

`int aal7_setMaxRecvWinSize()`

- Used to adjust AAL7's maximum receive-window size

`int aal7_dump_vc_table()`

- Causes switch to dump VC table and various statistics
- Really a gross violation of layering, but WE WANT TO SEE YOUR TABLES!