CSC 525:
Principles of Computer Networks

On Distributed Communications Networks

- by Paul Baran at RAND in early 60's
- one of the first proposals for packet switching and dynamic routing

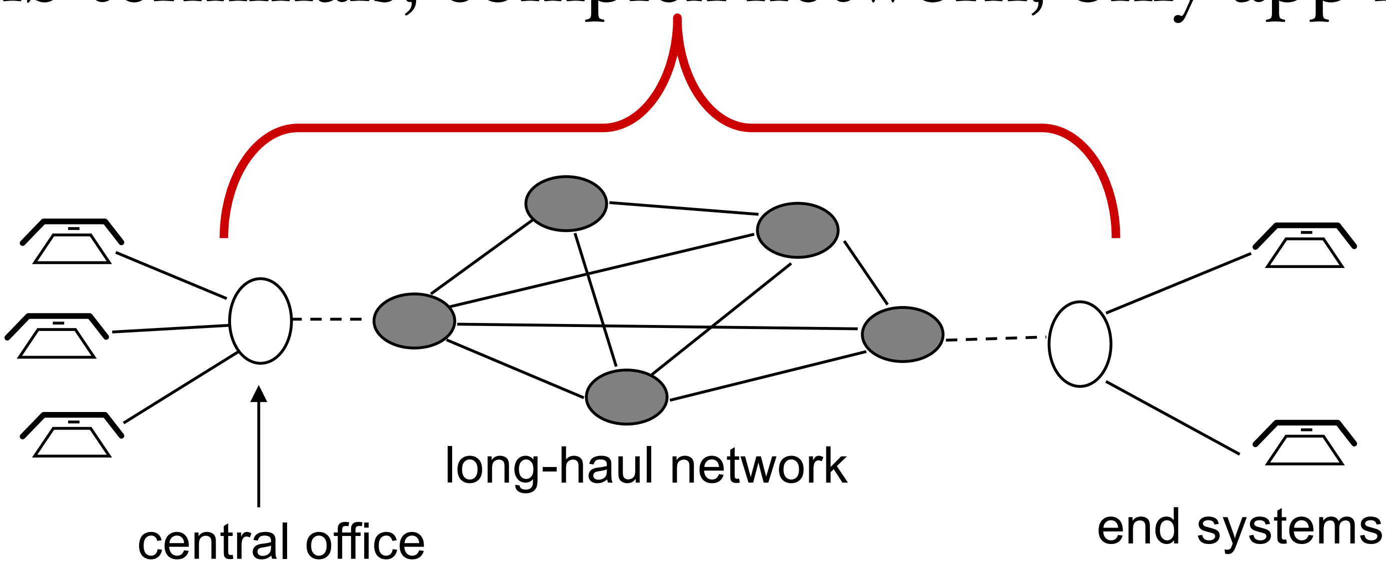
- To understand the fundamental reasons for packet switching and dynamic routing

Side Topic: Reading Papers

- First Pass:
  - What is the goal? What is the problem to solve?
  - What is the problem context?
    - Background, previous work, assumptions
  - What is the main idea of the work?
  - To accomplish this:
    - Carefully read abstract, intro, main design, and conclusions
    - Skim analysis/evaluation section
- Second Pass
  - Start with the assumption that the main point is wrong
    - (even if you agree with it :)
  - Focus on analysis/evaluation sections and the details of the design.
    - How do the analysis and results prove otherwise?
  - What are the weak points?

Telephone Networks

- Circuit switched, time division or frequency division.
- Centralized switching control
- System reliability relies on components' reliability
  - individual nodes are not expected to fail
- Dumb terminals, complex network, only app is voice.



IP Networks

- Packet switched, statistical multiplexing.
- Dynamic Routing
  - adapt to component failures and other changes in networks.
- Complex hosts, dumb network, built on redundant unreliable components to achieve system reliability.
- Supporting a wide range of applications
- All envisioned in Baran's paper

Building Robust Communication Systems

- To achieve high <u>system</u> availability in the presence of <u>component</u> failures
  - Redundancy in components (links and nodes)
  - Absence of centralized control
  - Cut data into packets, each packet carries its own destination address.
  - Dynamically route packets around failed parts
- Provide a common all-digital service for a wide range of applications.

Types of Networks

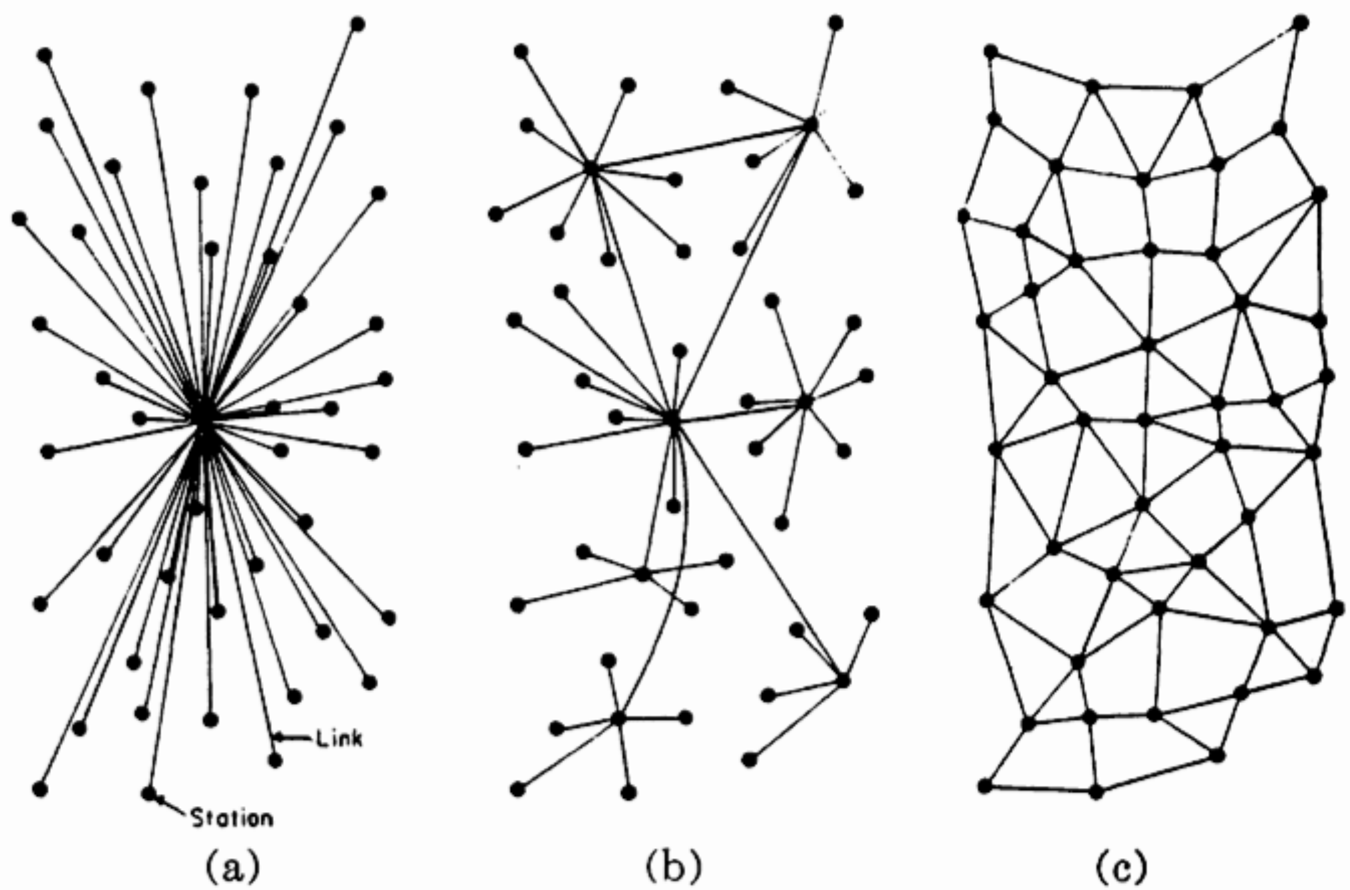


Fig. 1—(a) Centralized. (b) Decentralized. (c) Distributed networks.

Common Telco Designs Proposed Mesh

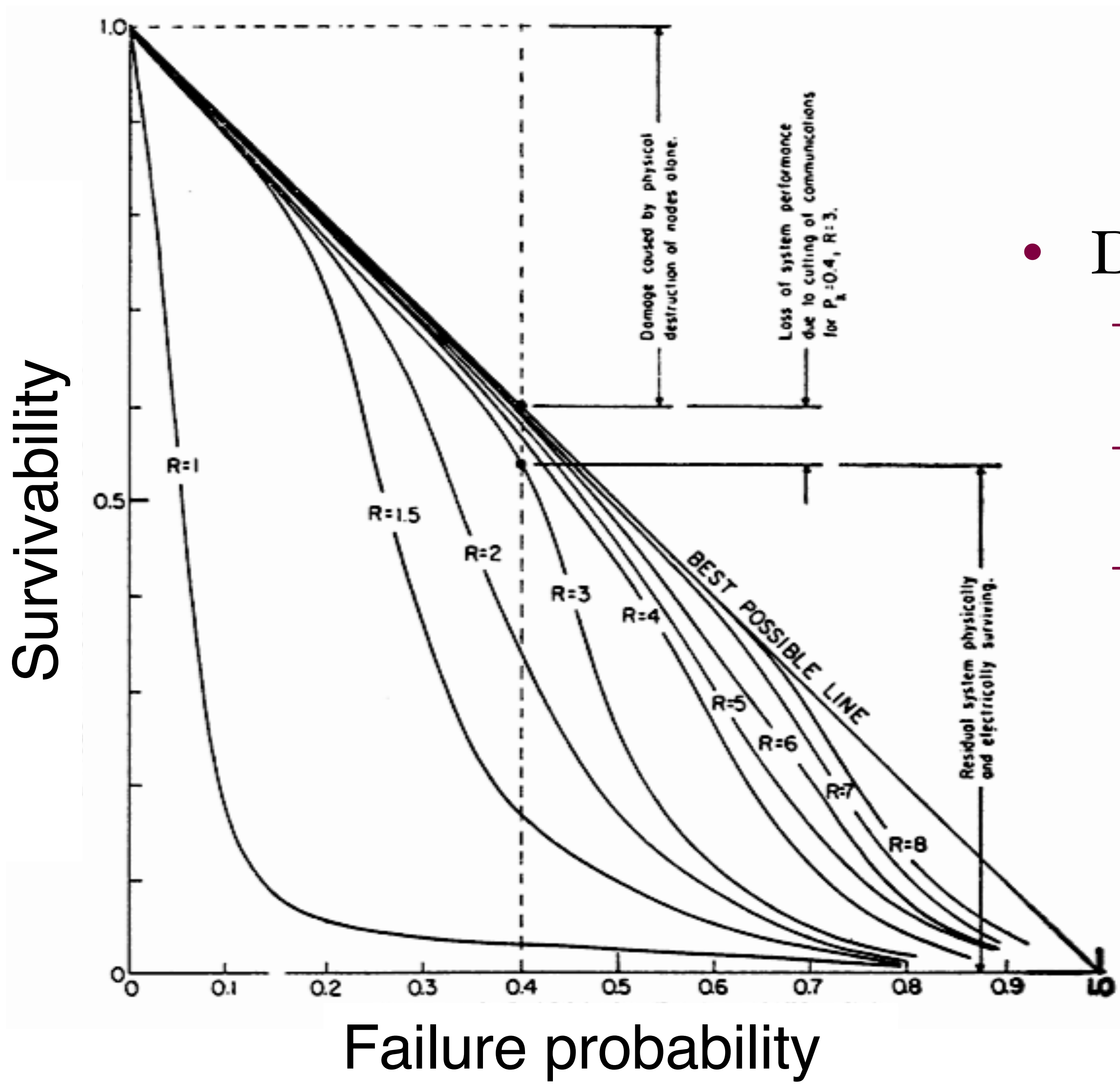
Network Redundancy

- Define redundancy level = #links/#nodes
- E.g., centralized star topology has redundancy level of 1.
  - A star topology has n nodes and n-1 edges.
- Best case is a full mesh (clique)
  - Every node connects to every other node.
  - But clique is expensive in number of links
  - Does not ***scale***

Side Topic: Scalability

- Being scalable is one of the most fundamental challenges in Internet protocol design.
  - How do you build a system that can grow to an arbitrary large size and still maintain its reliability and efficiency?
  - Many problems are easy in small scale, but hard in large scale.
- See how the design in each paper handles scalability issue.
- Another fundamental challenge is to handle dynamics/changes, especially unexpected ones.

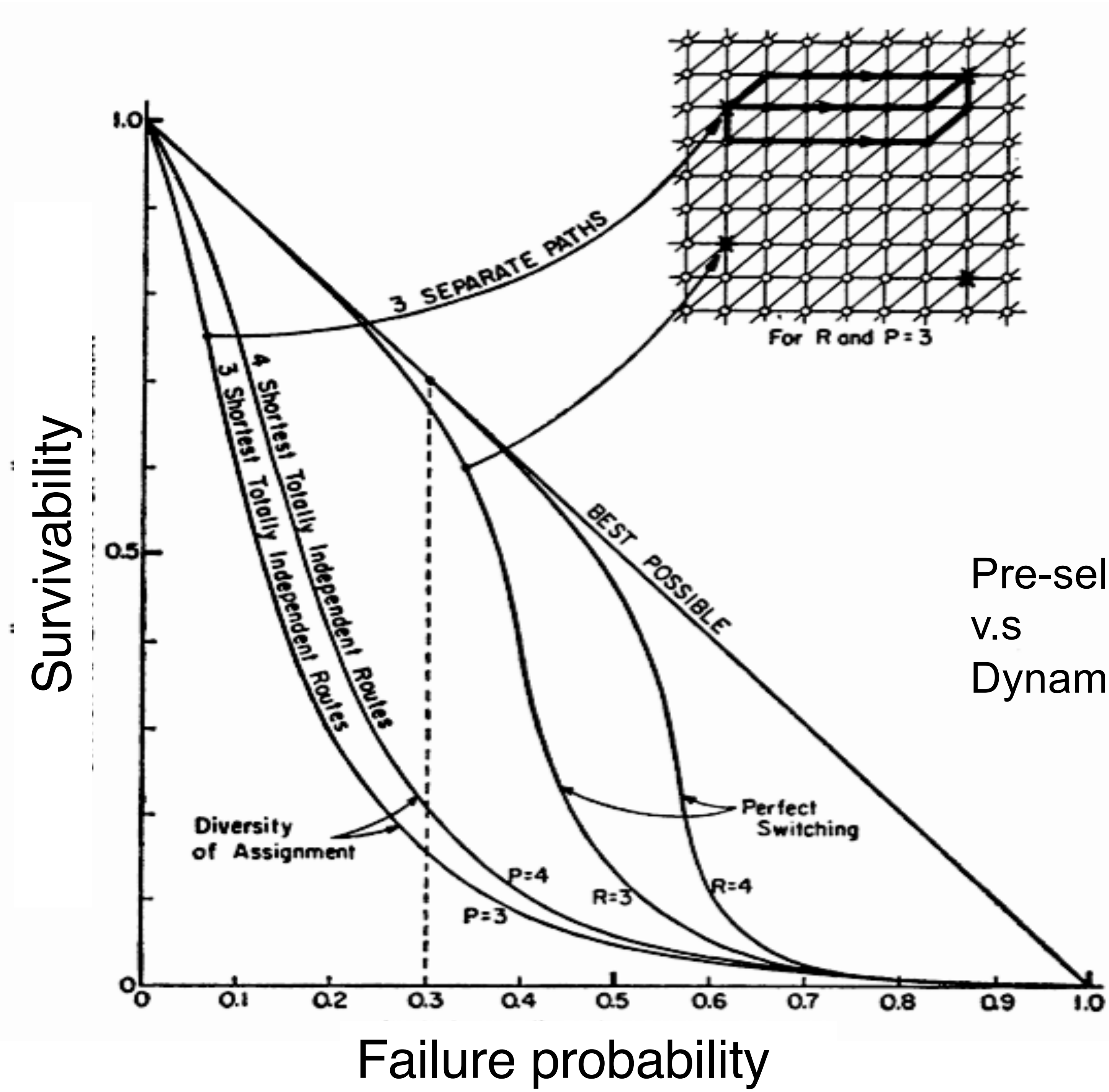
Survivability and Redundancy



- Do we need a full mesh?
  - If yes, solution is not realistic.
  - Result is that redundancy of 3-4 may suffice.
  - Further increase of redundancy doesn't help very much.

How to Select Routes

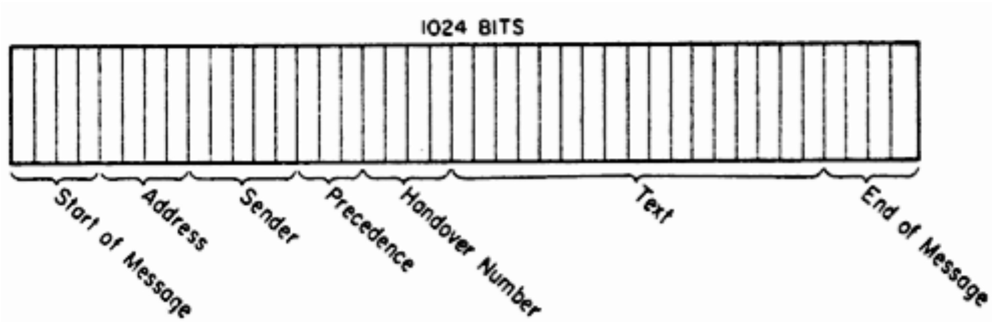
- Goal: To take advantage of link redundancy in case of failures.
- Options:
  - Pre-select a single route, clearly bad.
  - Pre-select several routes (diversity of assignment)
    - So no need for runtime path computation
  - Dynamic routing (perfect switching)
    - Will need to find alternative path upon failures



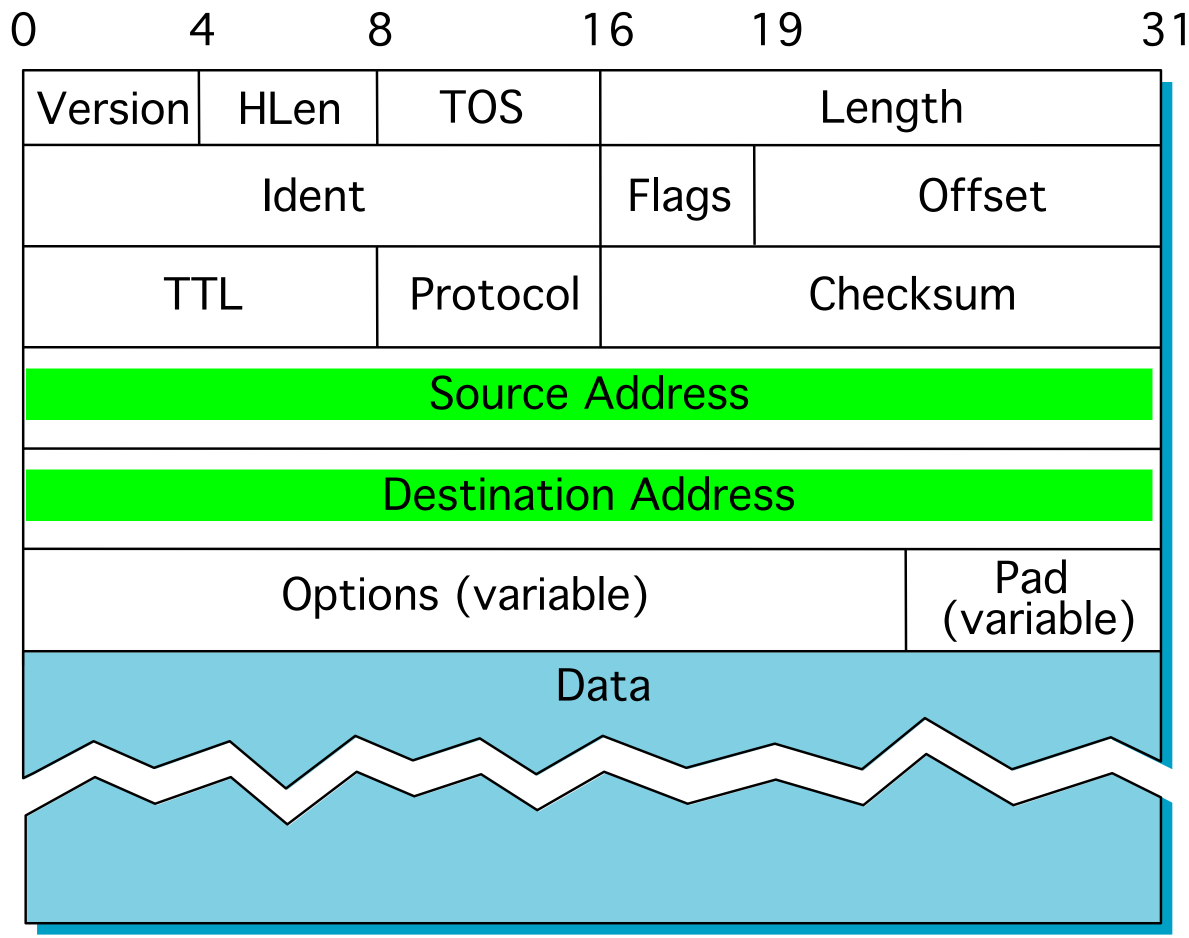
Pre-select Routes
v.s
Dynamic Routing

Packets

- A standard packet header format
  - source & dest. addresses, hop-count
- Store-and-Forward
  - concept of statistical multiplexing
  - different users/applications have different throughput requirements
  - send whenever packets are ready, no connection setup overhead.

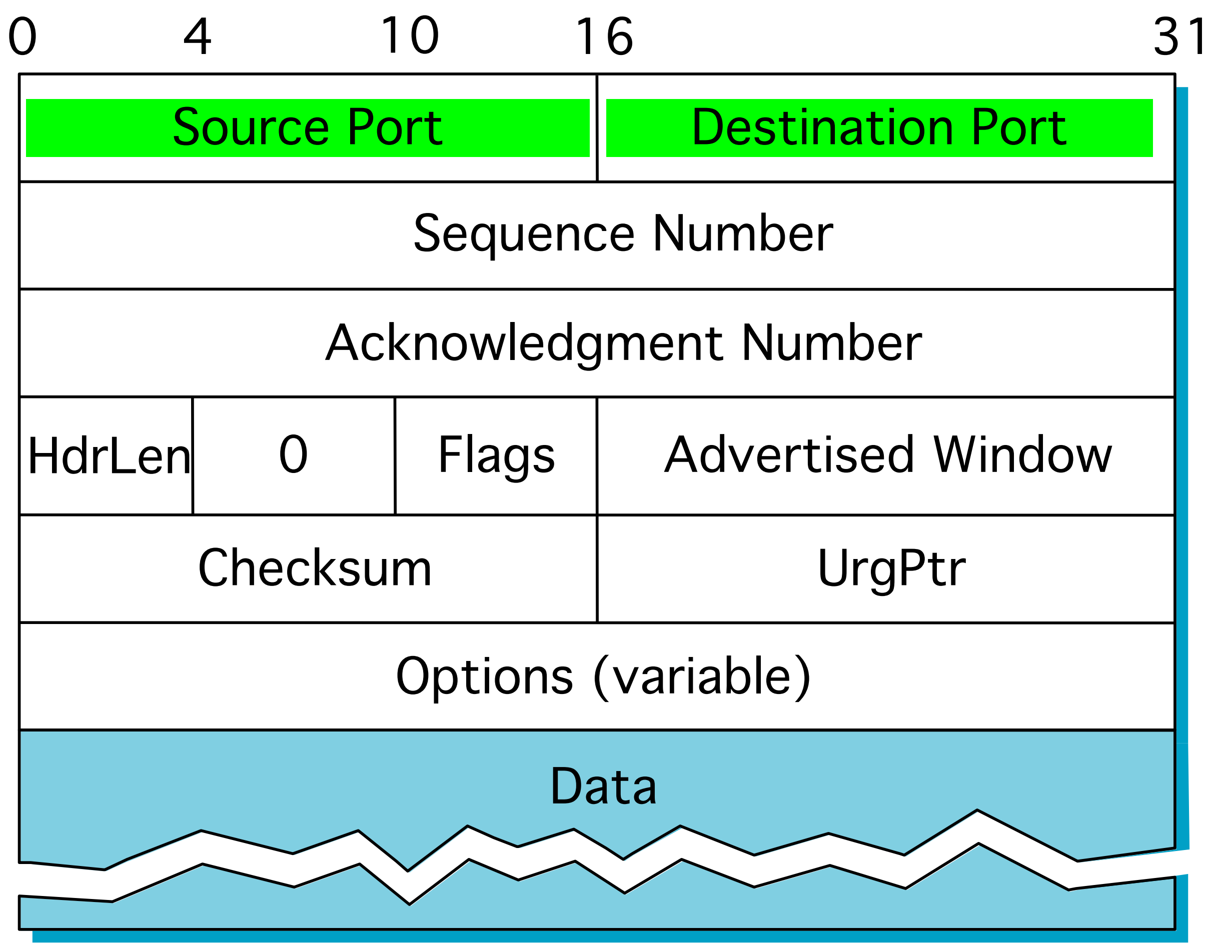


IP Header



Best effort packet delivery

TCP Header



End-to-end reliable byte stream

Dynamic Routing

- Interesting concept
  - Built a richly connected distributed network
  - Built packets with source and destination addresses
  - Claim we need dynamic routes
    - in order to get around failed links
- But how would you do it?
  - What about time, memory, storage, buffer, etc?
  - Today we have a number of routing protocols (RIP, OSPF, BGP etc.)

Hot Potato Routing

- Two equal objectives:
  - Send along shortest working path to destination.
  - Forward the packet as soon as you get it.
- Conflict: many packets may want to go out on the same interface for shortest path.
- Trade-off: send along the shortest *available* path

Routing Table and Forwarding Table

	LINK NUMBER							
	1	2	3	4	5	6	7	8
	HANDOVER NUMBER ENTRIES							
A	22	∞	12	10	9	9	8	13
B	5	3	2	2	4	5	12	2
C	7	8	13	9	22	10	7	8
D	21	23	19	21	12	10	12	13
E	7	10	12	14	12	13	13	15
F	7	10	12	13	14	2		
G	6	4	10					
Z	15	20	7	3	10	8	5	10

BEST CHOICE				
1st	2nd	3rd	4th	[illegible]
LINK NUMBER for DECISION CHOICE				
7	5	6	4	[illegible]
3	4	8	2	[illegible]
1	7	2	8	[illegible]
6	5	7	8	[illegible]
1	2	3	5	[illegible]
1	2	3	4	[illegible]
5	2	1	6	
4	7	3	6	[illegible]

Routing Table:
For each destination, maintain outgoing interfaces and distances

Forwarding Table:
For each destination, maintain the best outgoing interface.

Routing Lookup

Building the Forwarding Table

- Listen for incoming traffic, use hop count to estimate distance back to the source
  - Hop count is recorded in the packet header.
- When a router receives a packet
  - if the router has not seen the source address before, add it to the routing table
  - if the source already in routing table, and this packet has smaller hop count than the one in the table, update the table.
  - Otherwise do nothing.
  - Use info carried in data packets to build the routing table.
- Today's routing protocols use special packets, i.e., routing messages, to communicate and build the routing table.
  - Separate control and data.
  - But Ethernet uses a learning algorithm similar to this paper's.

Be Adaptive

- Need to be adaptive to changes
  - Component failures, malicious attacks, etc.
- Use exponential average to take present and past into consideration
  - H(next) = a*H(current) + (1-a)*H(measurement)
  - Tune it by adjusting the value of parameter a.
- Assign link weight to reflect the cost or distance of the link.
- These techniques are still being used today.

How to build a robust system

- Learn the changing environment by observations
- Automatically adjust to changes
  - Treat changes as norms, not exceptions
- With adequate redundancy and adaptivity, one can build a highly robust system out of unreliable parts
  - Vulnerable systems fail if any component fails
  - Robust systems fail only if all the parts have failed
  - A wide range in between
- One of the fundamental design principles in the Internet architecture

Summary

- The paper provides some context and history for network systems.
  - Suggests a move to dynamic packet switching networks rather than centralized circuits.
- Introduces some basic building blocks for Internet design
  - Distributed networks without centralized control
  - Standard packet header
  - Dynamic routing algorithm
- Keep these in mind in reading more recent papers and think how these are reflected in actual protocol design, and whether they fit well or not with emerging needs.