Tutorial - 7
ELEC - 4120

Manohar Kuse
mpkuse@ust.hk
http://ihome.ust.hk/~mpkuse

Topics for Today

- Cyclic Redundancy Check (CRC)
- Network Routing
  - Link State Algorithms (Dijkstra’s)
  - Distance Vector (Bellman-Ford’s)

Cyclic Redundancy Check (CRC)

Recall

Parity Bits for Error correction??

We added an extra bit based on number of 1s

(# of 1s even ⇒ ‘0’)

(# of 1s odd ⇒ ‘1’)

Example

1000 000 ⇒ 1000 0001

Reference : http://en.wikipedia.org/wiki/Parity_bit

How the error correcting code play in?

1. Sender appends some extra error checking bits after the data
   (or in the error checking field in the reader of a datagram)
2. Receiver receives this data and the error checking bits
3.
   a. If data and the received error checking bits and consistent then declare the packet is error free
   b. if data and the received error checking bits are not consistent then declare a packet is erroneous.

Cyclic Redundancy Check

Simple and Fast

Can detect multiple bits of error

Application

32-CRC ⇒ Ethernet network standard

Aviation; Train Communication Nets; Telecom Networks; Storage of PNG, MPEG, etc; USB data transfers

Reference : http://en.wikipedia.org/wiki/Cyclic_redundancy_check

How to compute it?

Need to understand the two stages

1. Generating the error bits
2. Checking the consistency in received data and its error bits

1. Generating Error bits

Note : This step is performed at sender

Inputs:

a. *Data*;

eg. 1000 1010

b. *Generator Polynomial*;

eg. x^3 + x^2 + 1 ie. 1101. Note the degree of the polynomial is 3

Output:

a. *Error checking bits*. Number of error checking bits will be 3 in this case (equals the degree of polynomial)

Generating Error bits - Illustration

Data : 11010011101100

Generator Polynomial : x^3+x+1, ie. 1011

Explanation on the board.

Note : This example is borrowed from wikipedia page on CRC.

Refer : http://en.wikipedia.org/wiki/Cyclic_redundancy_check#Computation_of_CRC

2. Checking Validity of Received Message

Inputs:

a. Received data with error checking bits

b. Generator Polynomial : x^3+x+1, ie. 1011.

Note that this polynomial is same as the one used at the sender

Output:

Tell us if the message is valid or invalid (erroneous)

Checking Validity of Received Data - Illustration

Received Data : 110100111101100 100

Explanation on the board.

Note : This example is borrowed from wikipedia page on CRC.

Refer : http://en.wikipedia.org/wiki/Cyclic_redundancy_check#Computation_of_CRC

Online CRC Calculator

http://ghsi.de/CRC/index.php?Polynom=10011&Message=35B

Routing

What is Routing

Selecting the best path for ‘Sending the packet from source to destination’.

Think from the basics

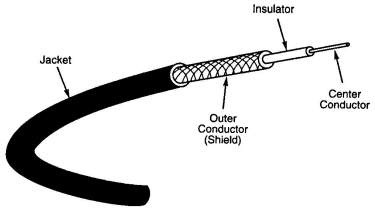
Lets imagine situation as in 1980s.

Entire university had 1-2 computers

We have a program to send packet to only its neighbour

Simple Networks

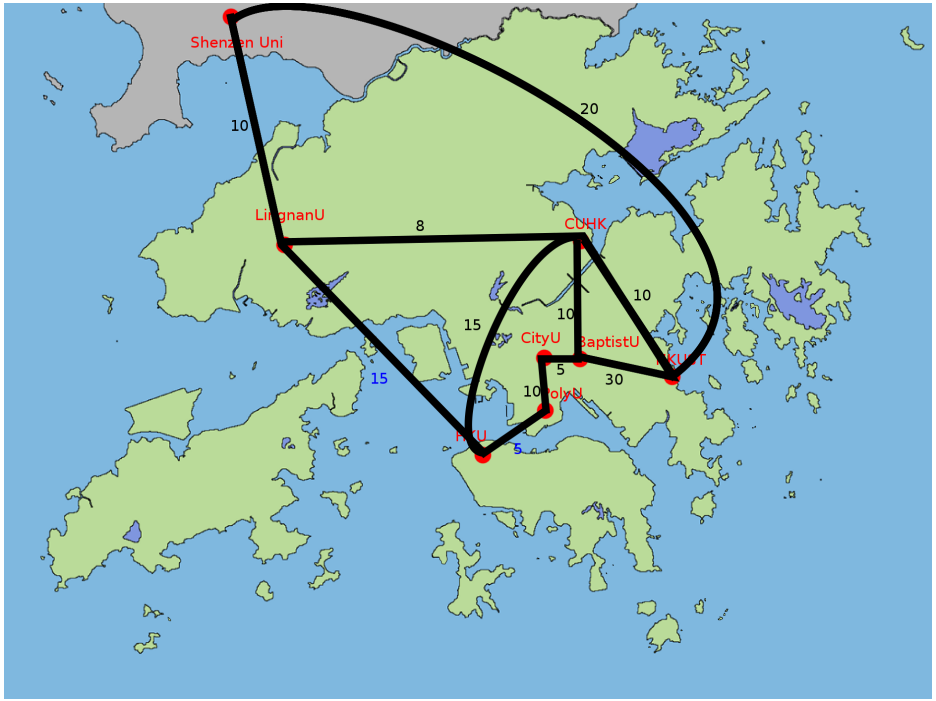
Computers between universities are connected with co-axial cables



Example Network Layout

Lay the wires
like these

Number proportional
to time taken across
the link



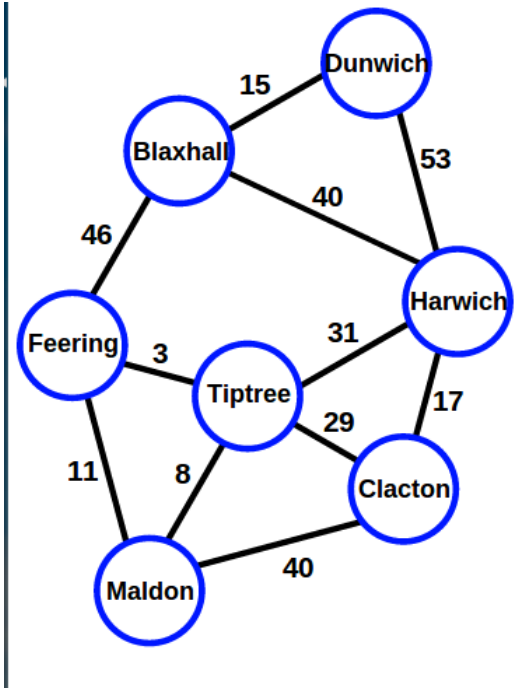
Graph

Vertices & Edges

Weighted Undirected graph

Refer :

http://en.wikipedia.org/wiki/Graph_(mathematics)



Shortest Path Algorithms in Graphs

Dijkstra’s Algorithm (Link-State)

Bellman-Ford’s Algorithm (Distance Vector)

Dijkstra's Algorithm Illustration

Input:

- Graph (This can be represented as a matrix)
- Source

Output:

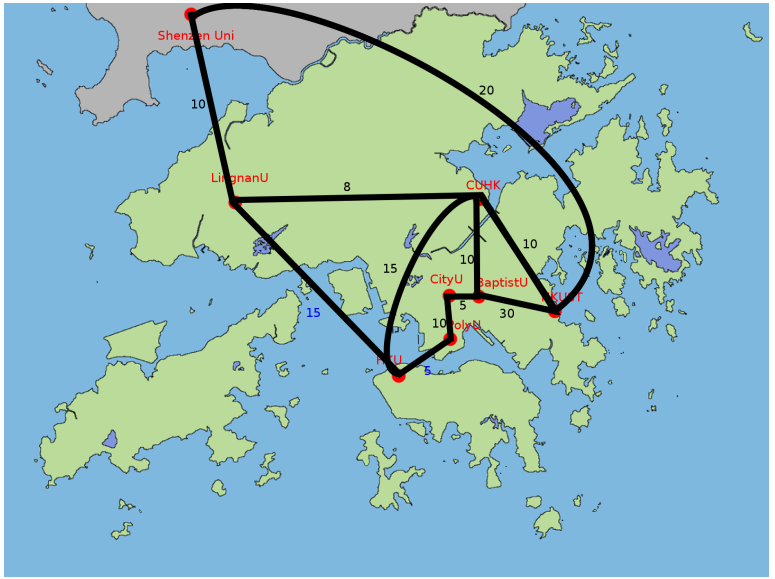
- Shortest paths to all the destinations

Dijkstra's Algorithm Illustration

Toy HK Network

demonstrated on board.

See youtube video for ref.



Refer:

http://en.wikipedia.org/wiki/Dijkstra's_algorithm (See the animation)

http://www.youtube.com/watch?v=WN3Rb9wVYDY

Systematically Presenting the output (Exams only)

1. Fill up such a table

Iteration #	Visited	Current	A	B	C	D	E	F	G	H

2. Backtrack paths to all destination

A to B ; A to C ; A to D ; A to E ; A to F ; A to G ; A to H

Iteration #	Visited	Current	A	B	C	D	E	F	G	H
0			0, N/a	inf	inf	inf	inf	inf	inf	inf
1		A		10,A		20,A				30,A
2	A	B			18,B		25,B			20,B
3	AB	C								
4	ABC	D								
5	ABCD	H							25,H	
6	ABCDH	E						30,E		
7	ABCDHE	G								
8	ABCDHEG	F								
final	ABCDHEGF		0,N/a	10,A	18,B	20,A	25,B	30,E	25,H	20,B

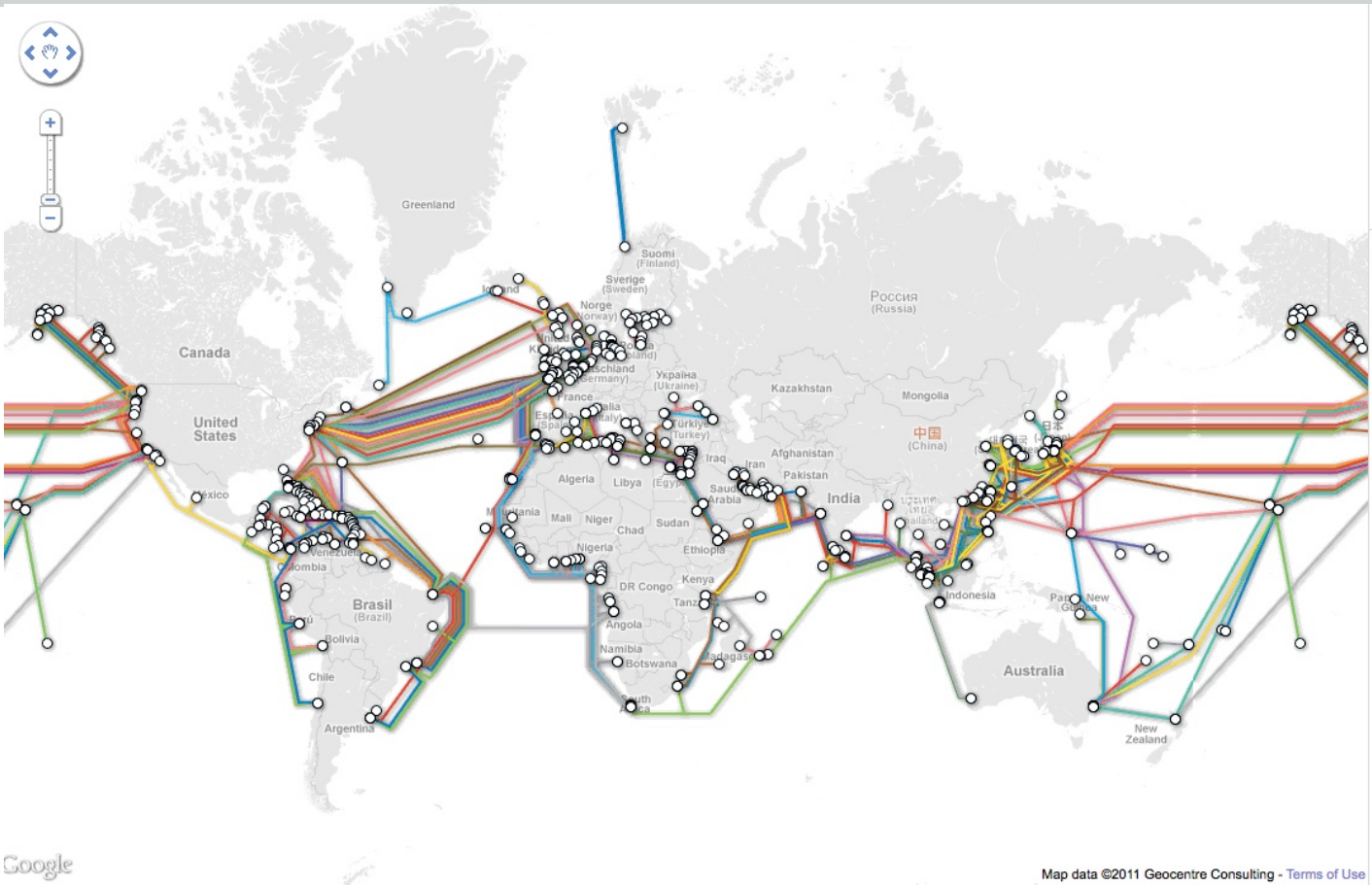
Problems with Dijkstra's Algo

Such computations has to be done at each node

Need to have full info about the network topology by each node wishing to send data.

Centralized

Not adaptive to changing weights



World Undersea Cables

Full List & More interesting info : http://www.submarinecablemap.com/

Bellman-Ford's Algo

Pros:

a. Uses only info about neighbours.

b. Decentralized. Suits better for network.

Cons:

a. Very Slow

b. Not adaptive

Self Study Bellman-Ford

Watch these videos in this sequence:

1. Gives necessary background

   https://www.youtube.com/watch?v=W2fKGISUAtM

   https://www.youtube.com/watch?v=sM7QAS39ZZ4

   https://www.youtube.com/watch?v=Tbw9eAot8hk

2. Example. This video is probably most important to watch

   https://www.youtube.com/watch?v=ZGaqiEgy97I

Routing in Modern Internet

Short Answer :

BGP (Border Gateway Protocol)

Know more : http://en.wikipedia.org/wiki/Border_Gateway_Protocol

Both algorithms are not very good at a large scale and have their own problems

Use Hybrid

Routing in Modern Internet

Hybrid Hierarchical Routing

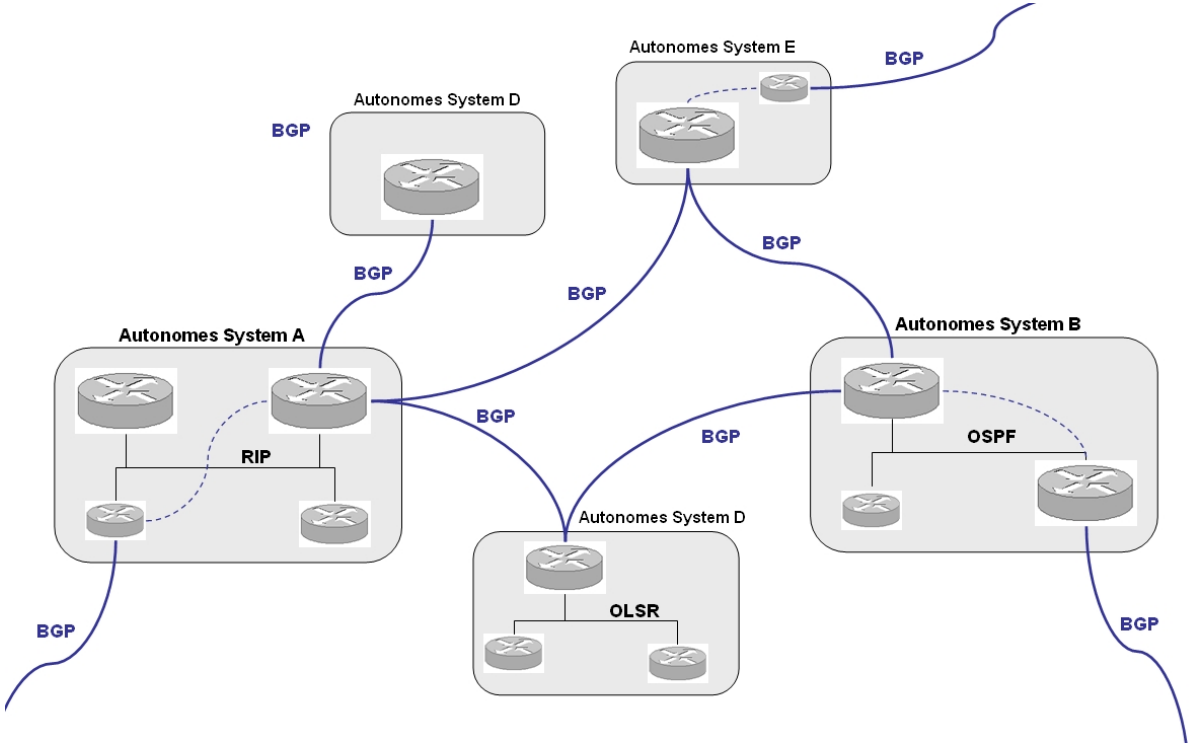
Understand these terms:

a. AS (Autonomous Systems)

b. Intra-AS Routing

c. Inter-AS Routing

d. Gateway router


Autonomes System E
BGP
Autonomes System D
BGP
BGP
BGP
BGP
Autonomes System A
Autonomes System B
BGP
BGP
RIP
OSPF
Autonomes System D
OLSR
BGP
BGP

Intra-AS Routing

1. Each AS is free to run its own routing
2. These intra-AS routing algorithms can be Link-state or Distance Vector.
3. Strategies to adapt to changing weights

Standard Intra-AS routing

1. OSPF - http://en.wikipedia.org/wiki/Open_Shortest_Path_First
2. RIP - http://en.wikipedia.org/wiki/Routing_Information_Protocol

Inter-AS routing

Single point of entry to the network is the gateway router.

All communications should happen with the Gateway router

Routing between these gateway router happens with **BGP**

http://en.wikipedia.org/wiki/Border_Gateway_Protocol