Wireless Ad Hoc Networks Lab 7

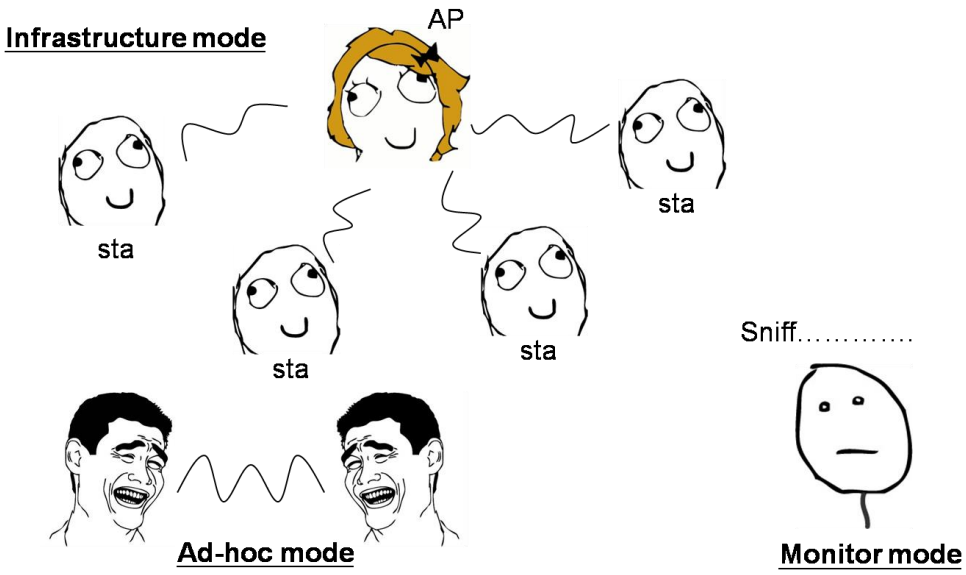
Ad Hoc Mesh Network

Wireless Operating Modes

Wi-Fi modes of operation (802.11 or Wi-Fi)

- Station (STA) infrastructure mode
  - This mode is also called **“ Managed “**
- AccessPoint (AP) infrastructure mode
- Monitor (MON) mode (i.e, Sniff mode)

  Don’t need to connect any AP
- **Ad-Hoc (IBSS) mode**



Wireless NIC on Ad-Hoc mode

- An **IBSS (Independent Basic Service Set)** network,
  - often called an **ad-hoc** network
  - a way to have a group of devices talk to each other wirelessly, without a central controller
  - All devices talk directly to each other, with no inherent relaying
- How create a new interface (Linux iw command)
  - sudo iw phy phy0 interface add adhoc0 **type ibss**
- How to join an adhoc network (Linux iwconfig command)
  - sudo iwconfig adhoc0 **mode ad-hoc** essid **bun-mesh** channel **11**

Set wireless NIC on Ad-Hoc mode (1)

Before set NIC to adhoc mode, remember to stop/disconnect all the wireless network connection

- turn off the interface
  - sudo ifconfig wlan0 down
  - sudo iw dev wlan0 del
- create a new interface
  - sudo iw phy phy0 interface add adhoc0 type ibss

Set wireless NIC on Ad-Hoc mode (2)

- join the adhoc network
  - sudo iwconfig adhoc0 **mode** ad-hoc **essid** bun-mesh **channel** 11
- configure the IP address
  - sudo ip -6 addr add FE80::42:42:4x/128 dev adhoc0 (for IPv6)
  - sudo ip addr add 192.168.10.4x/32 dev adhoc0 (for IPv4)
- turn on the interface
  - sudo ifconfig adhoc0 up
- check the wireless interface status
  - sudo iwconfig

Babel Routing Protocol (1)

- **Babel** – a loop-avoiding distance-vector routing protocol
  - Based on ideas in DSDV, AODV, and Cisco's EIGRP
  - Designed to work well not only in wired networks but also in wireless mesh networks
  - Is in the process of becoming an IETF Standard

- Resources:
  - https://www.irif.fr/~jch/software/babel/
  - https://tools.ietf.org/html/rfc6126 (RFC Draft)
  - https://github.com/jech/babeld (Source Code)

Babel Routing Protocol (2)

- Loop-avoiding
  - Uses distributed Bellman-Ford
  - Feasibility condition guarantees good transient behavior
- Metrics
  - Hop-count on wired links
  - ETX (packet loss) on wireless links
  - Babel-Z3 (refines ETX by taking radio interference into account)
  - Babel-RTT (uses delay as component of routing metric)
- Route Selection
  - Choose route with smallest metric
  - Prefer stable routes

Babel Routing Protocol (2)

- Packets are sent in the body of a UDP datagram
- Hello Message
  - For Neighbor discovery
  - Broadcast periodically with a seqno (sequence number)
- IHU (I Heard You)
  - To confirm bidirectional reachability
  - Sent less often than Hellos
  - Carry the link's rxcost (reception cost)
- Route Request
  - Prompts receiver to send an update for a given prefix

Install Babel(1)

- Get babel from source
  - git clone git://github.com/jech/babeld.git
- Install
  - cd babeld
  - sudo -s
  - make
  - make install

Install Babel(2)

- Install directly from Ubuntu packages
  - apt-get install babeld
- run babel routing daemon
  - sudo babeld adhoc0
- Or, run babel routing daemon with debugging
  - sudo babeld –g 33123 adhoc0
  - telnet ::1 33123 (in another window/tab to observe)

Babelweb2 – Monitoring tool for Babel routing daemon

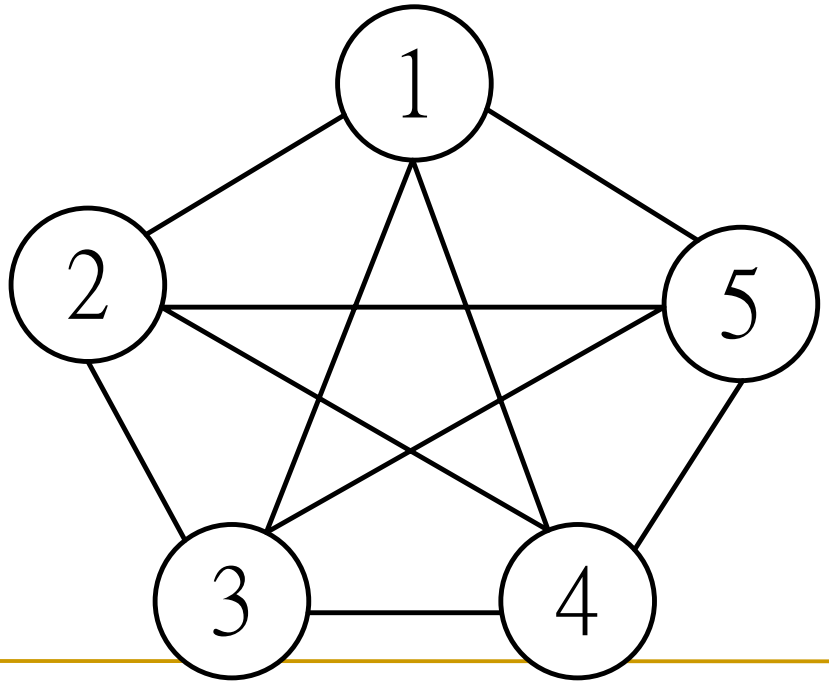
- Install dependencies
  - sudo apt-get install gccgo-go
  - wget https://dl.google.com/go/go1.13.5.linux-amd64.tar.gz
  - sudo tar -C /usr/local -xzf go1.13.5.linux-amd64.tar.gz
  - export PATH=$PATH:/usr/local/go/bin
  - export GOPATH=$HOME/golang
  - sudo rm /usr/bin/go
  - sudo ln -s /usr/local/go/bin/go /usr/bin/go

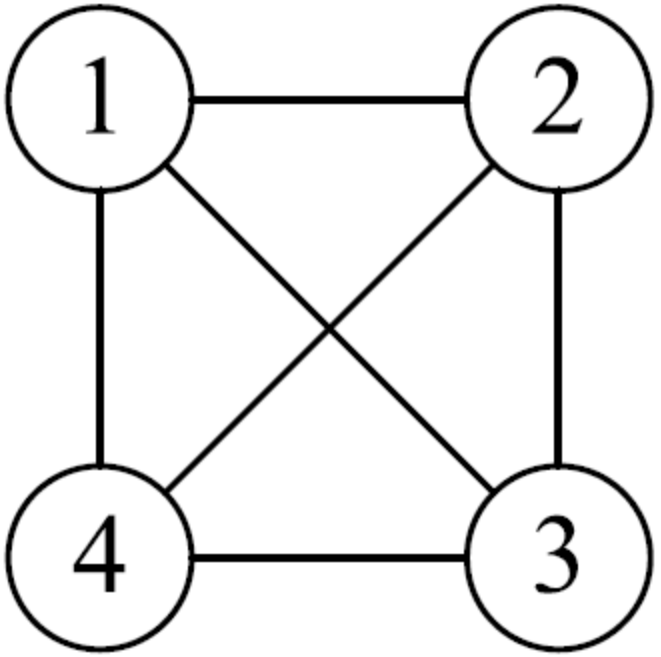
Babelweb2 – Monitoring tool for Babel routing daemon

- Get the repository from source
  - go get github.com/Vivena/babelweb2
- Install
  - go install github.com/Vivena/babelweb2
- Run babelweb
  - go run main.go
- Open browser to access Babelweb interface
  http://localhost:8080/

Create topology

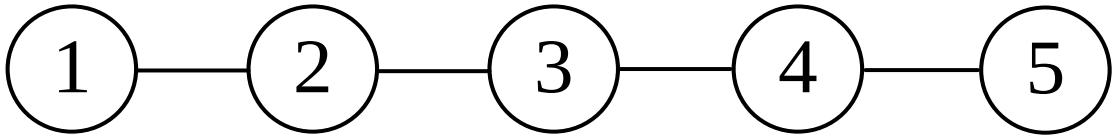
- When every node can connect to each other, the topology is a star-shaped topology.
- How can we create the desirous topology in a small space?
  - (ex: lab, classroom, office)





Create topology

- You can set the firewall rules to filter the packets from a specific node
- We can use the firewall rules to create a chain topology in a small space



Create topology

- firewall rules
  - Node ① will drop packets from ③, ④, ⑤.
    - Then node ① will only connect to node ②
  - Similar, node ② will drop the packets from ④, ⑤.
  - Repeat these rules on node ③, ④, ⑤.
  - Finally, node ① can communicate node ⑤ via node ②, ③, ④.

Create topology

- firewall rules (IPv6 address)
  - ip6tables -F
  - ip6tables -t filter -A INPUT -s xx:xx:xx:xx:xx:xx -j DROP
    - Fill in the IPv6 address
  - Use “**ip6tables –L**” to look up the current firewall rules.
  - Use “**ip6tables –F**” to clear the firewall rules

Experiment Part 1

Part 1

1. set NIC to adhoc mode
2. open Wireshark
3. Run babeld and babelweb
4. Observe the mesh network

Hint :

Use wireshark to observe babel packets

Use telnet to observe babel messages

Use babelweb to observe neighbor nodes & routing table

Use ping and ping6 to send packets to neighboring nodes

Experiment Part 2

Part 2

1. set the firewall rules
2. Observe the change

Hint :

Use babelweb to observe **change** in neighbor nodes & routing table

Use traceroute to observe the **multihop** traffic path