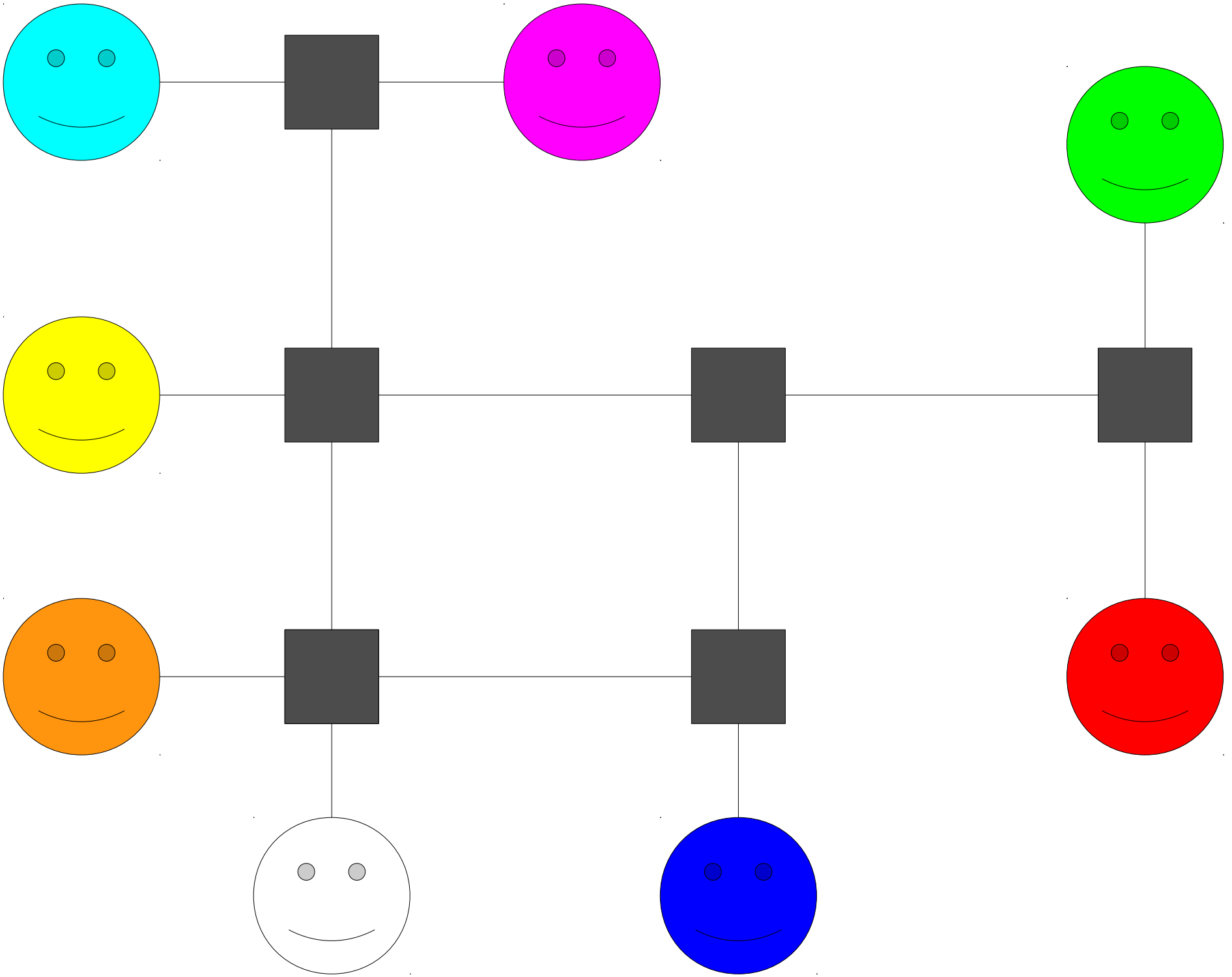
Networking

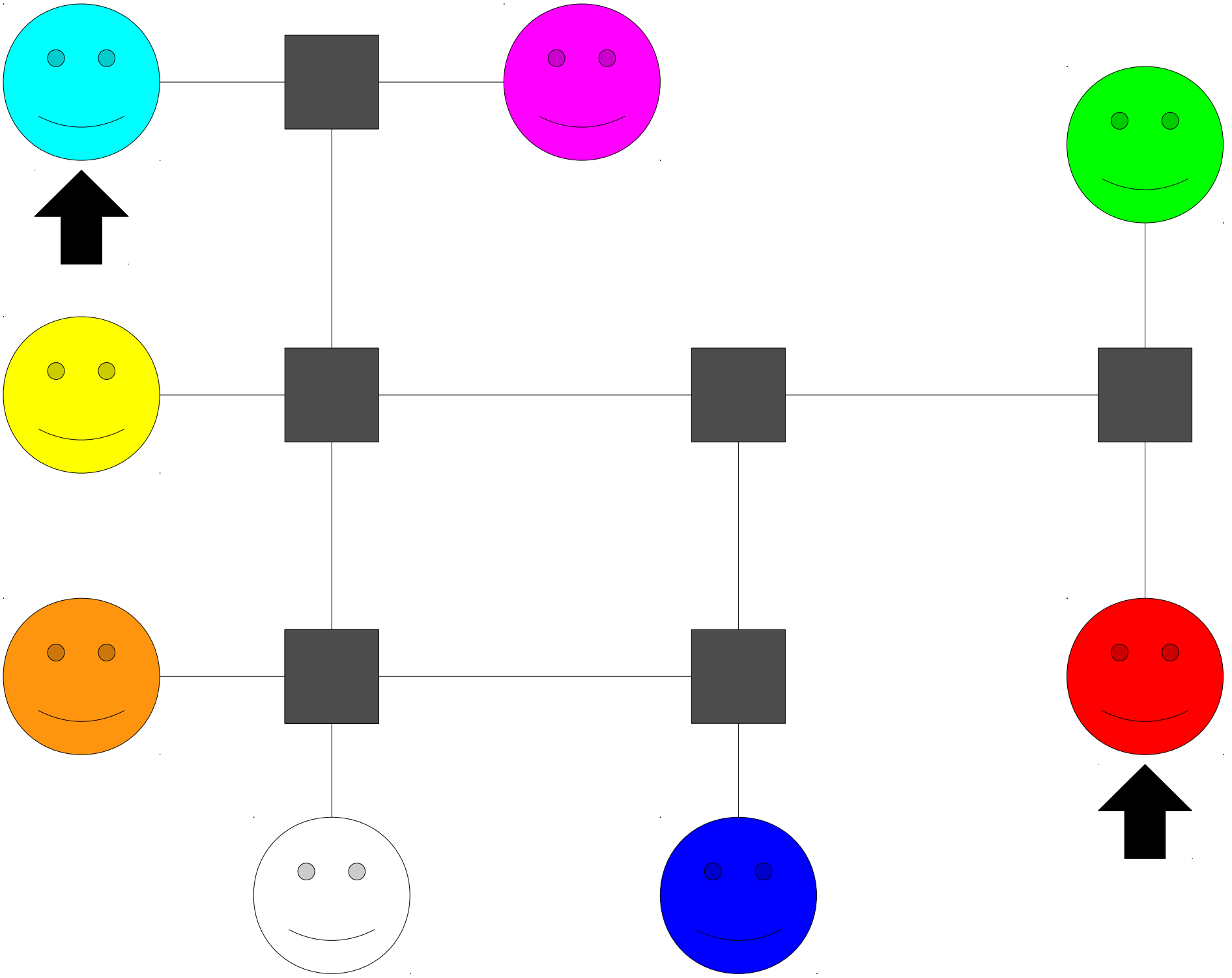
Friday Four Square!
Outside Gates, 4:15PM

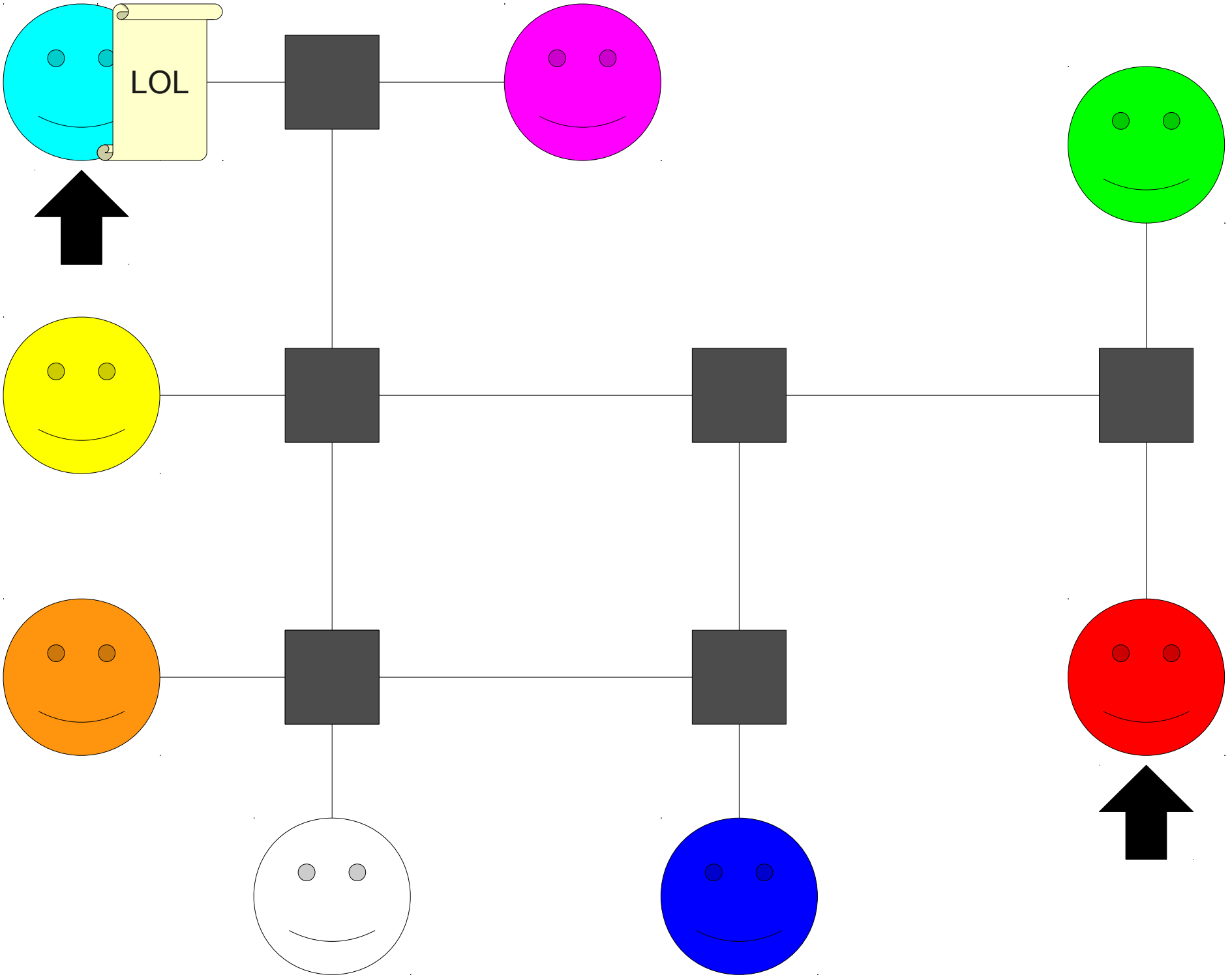
Computer Networks

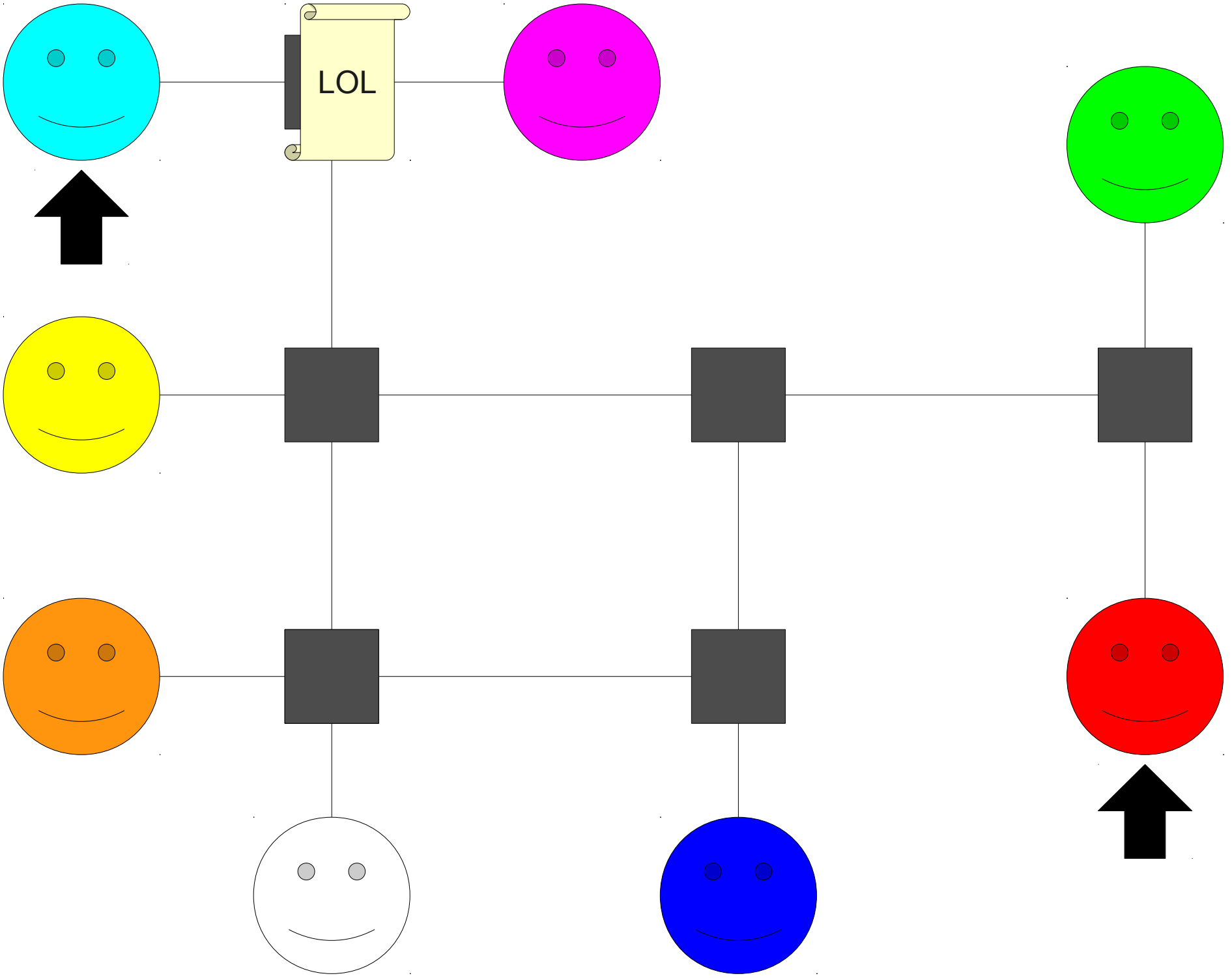
- Computer networks allow us to get amazing things done.
  - Sharing knowledge (Wikipedia, Khan Academy, etc.)
  - Solving huge problems (folding@home, SETI, etc.)
- Computer networks prevent us from getting amazing things done.
  - Social networks (Facebook, Google+, etc.)
  - Streaming video (Hulu, Netflix, etc.)

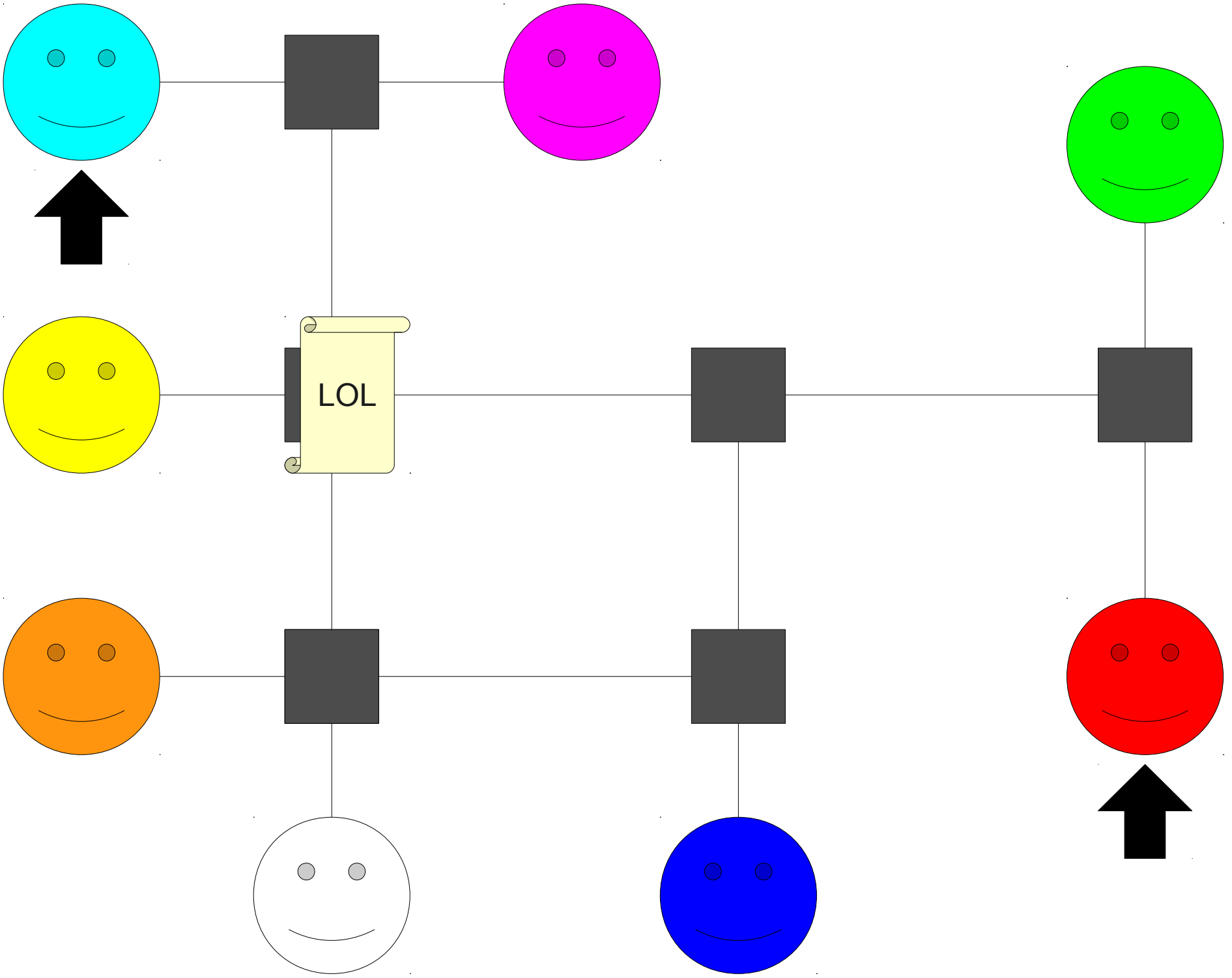
How does it all work?

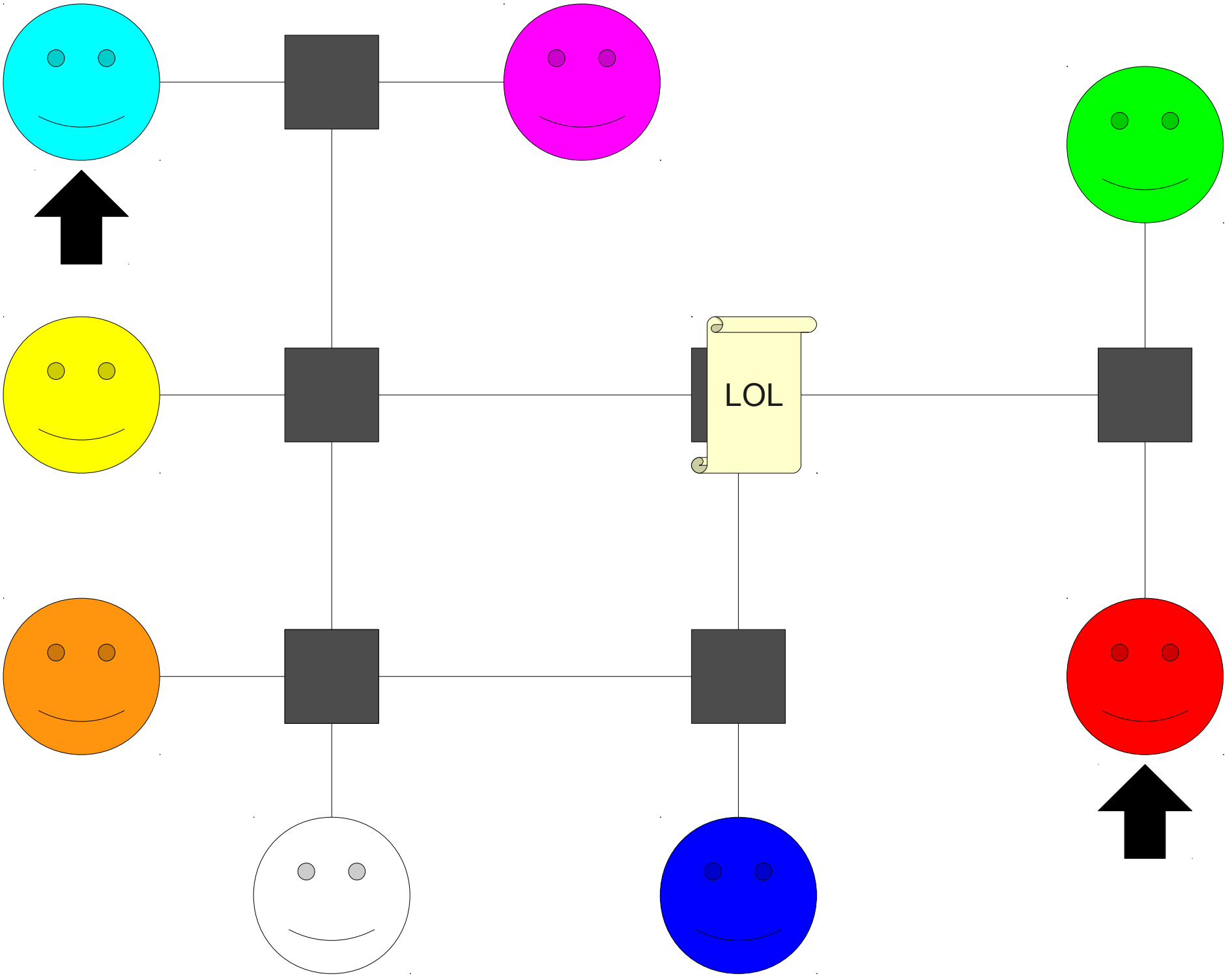


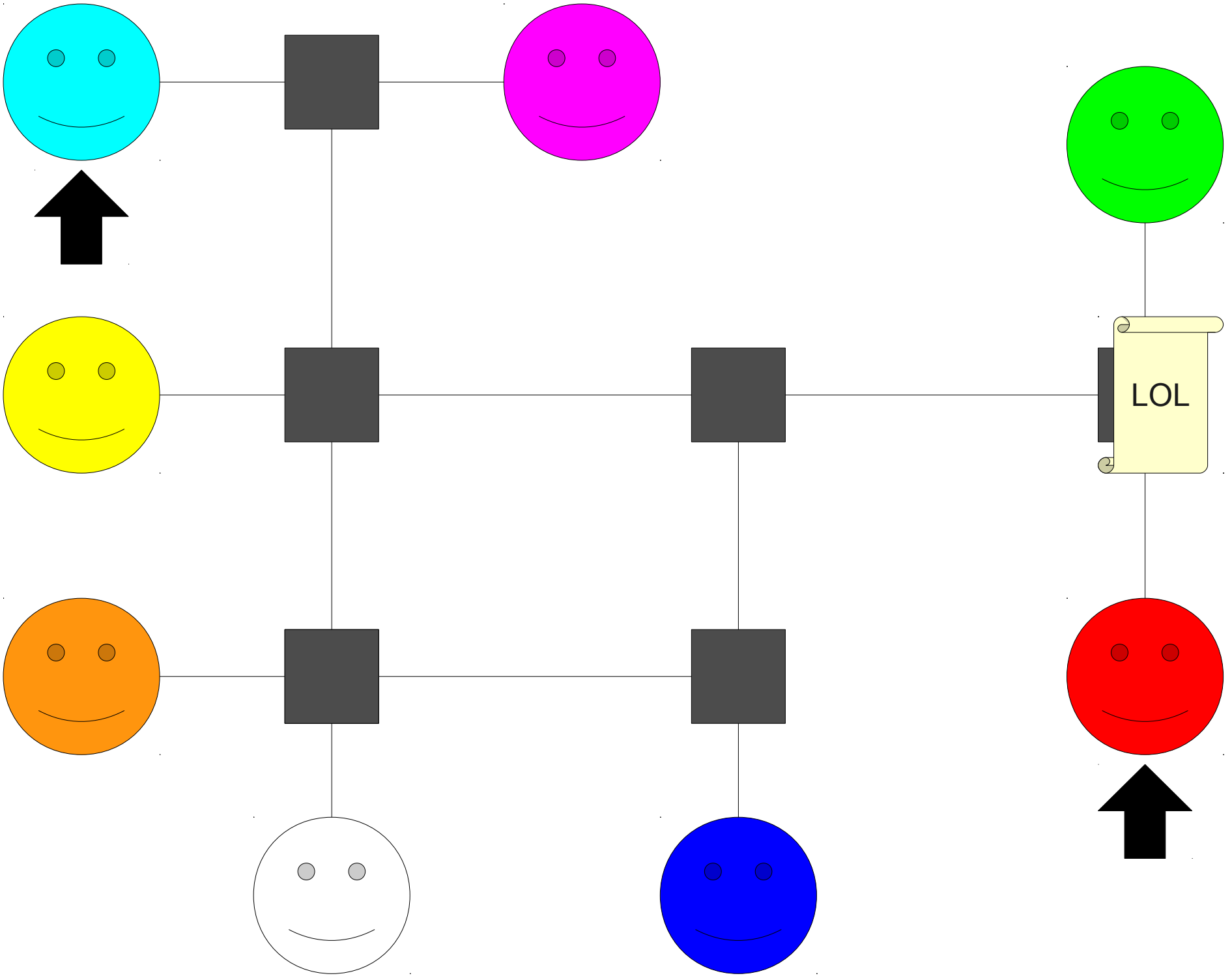


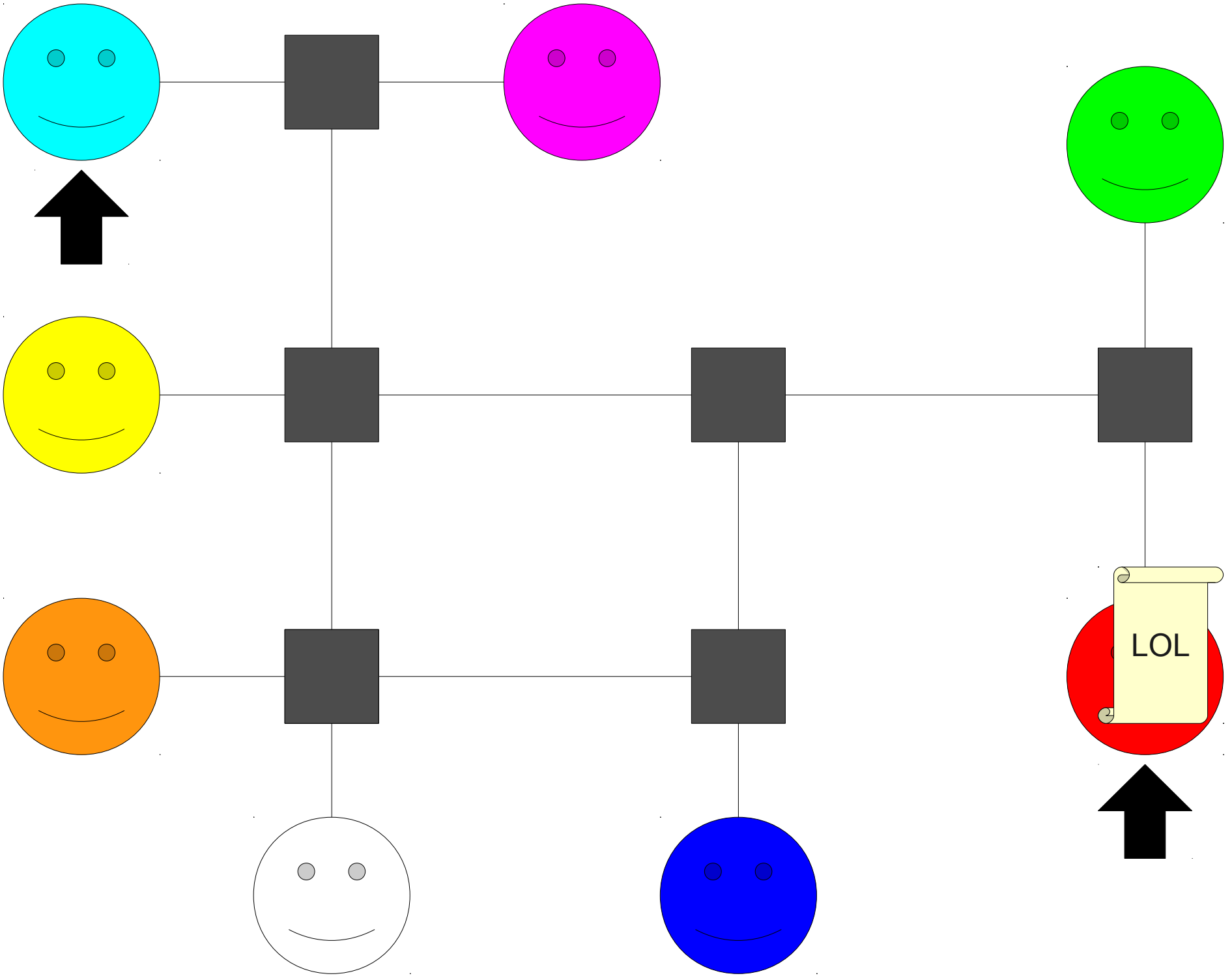

LOL

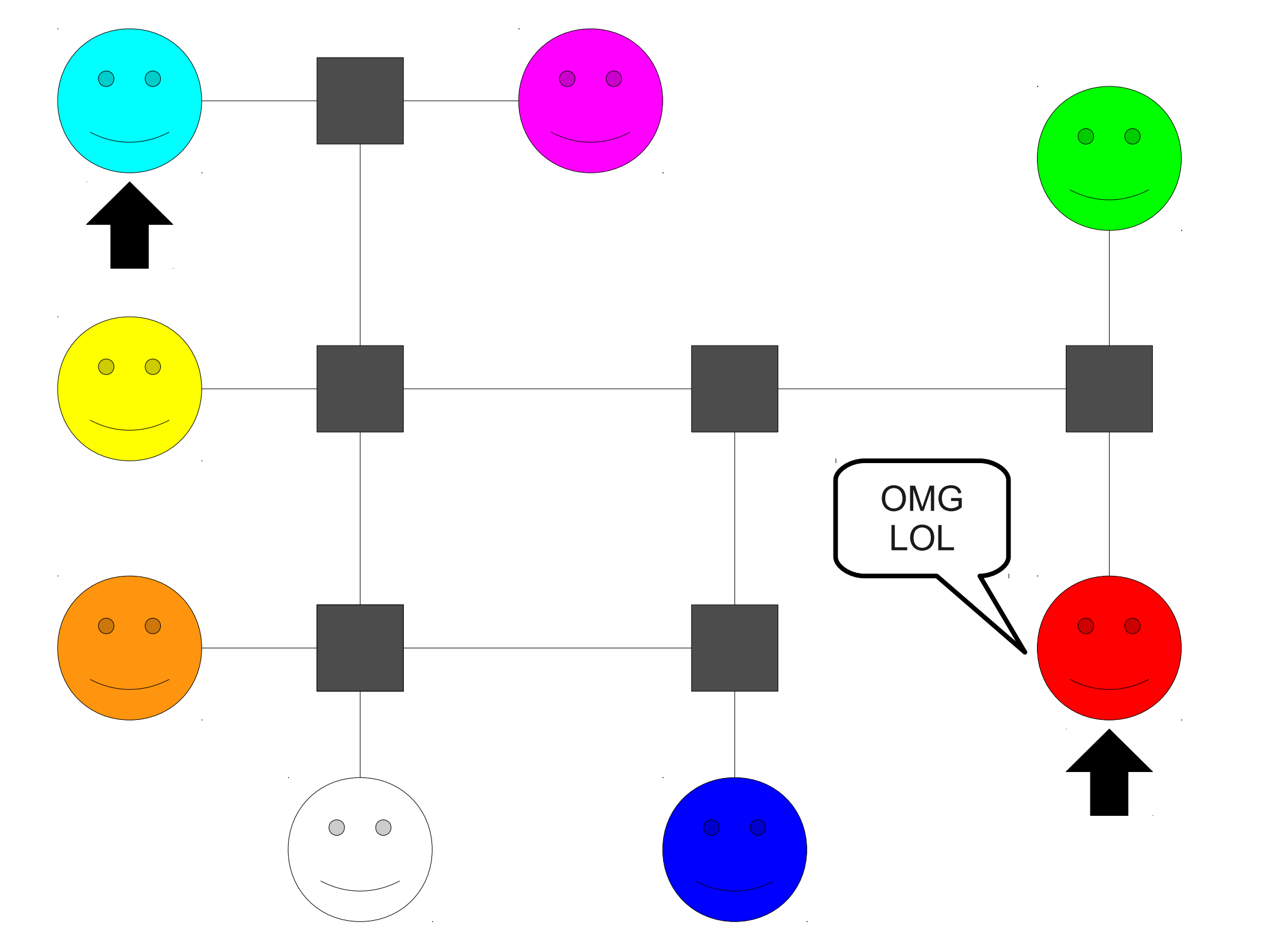

LOL


LOL


LOL

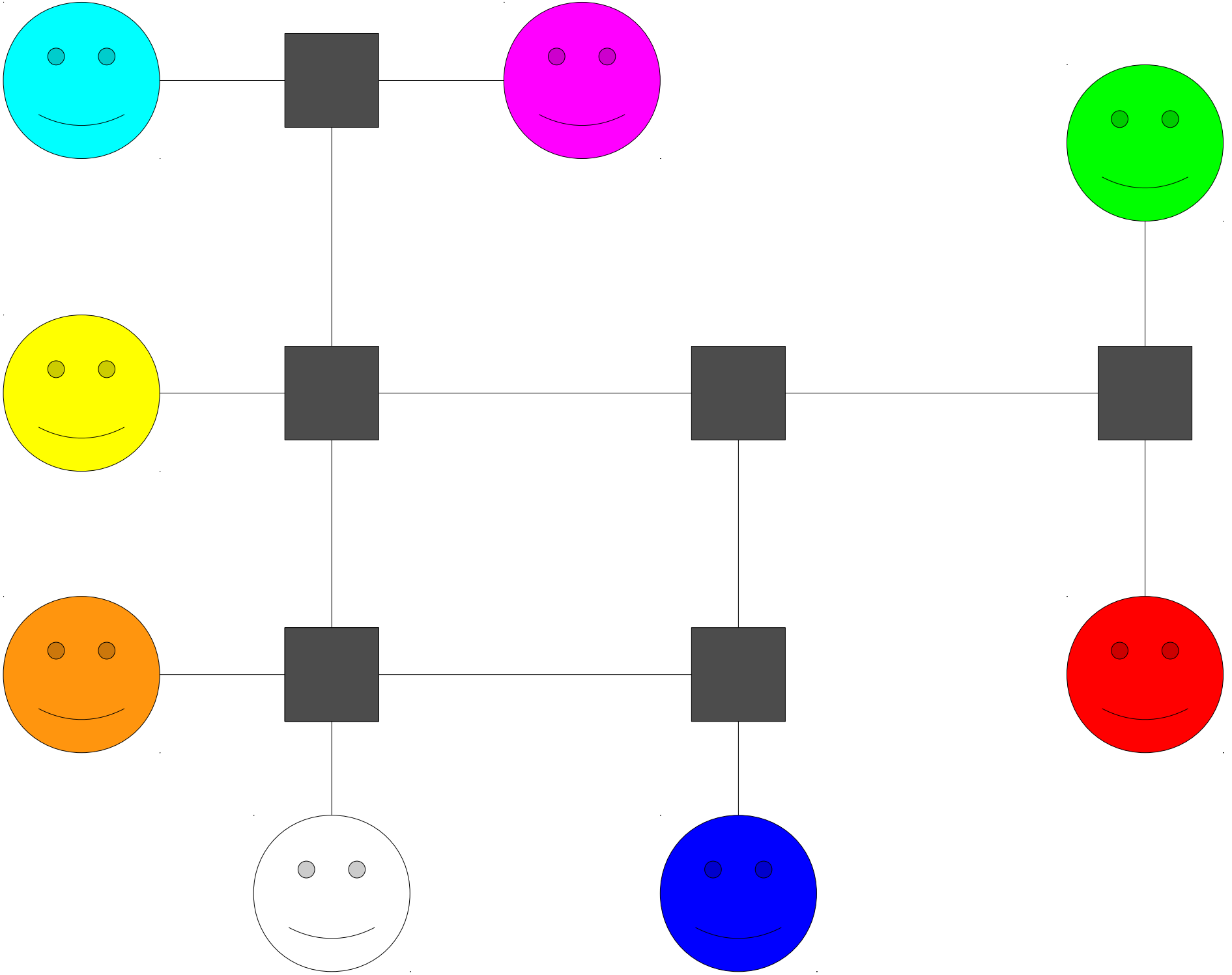

LOL

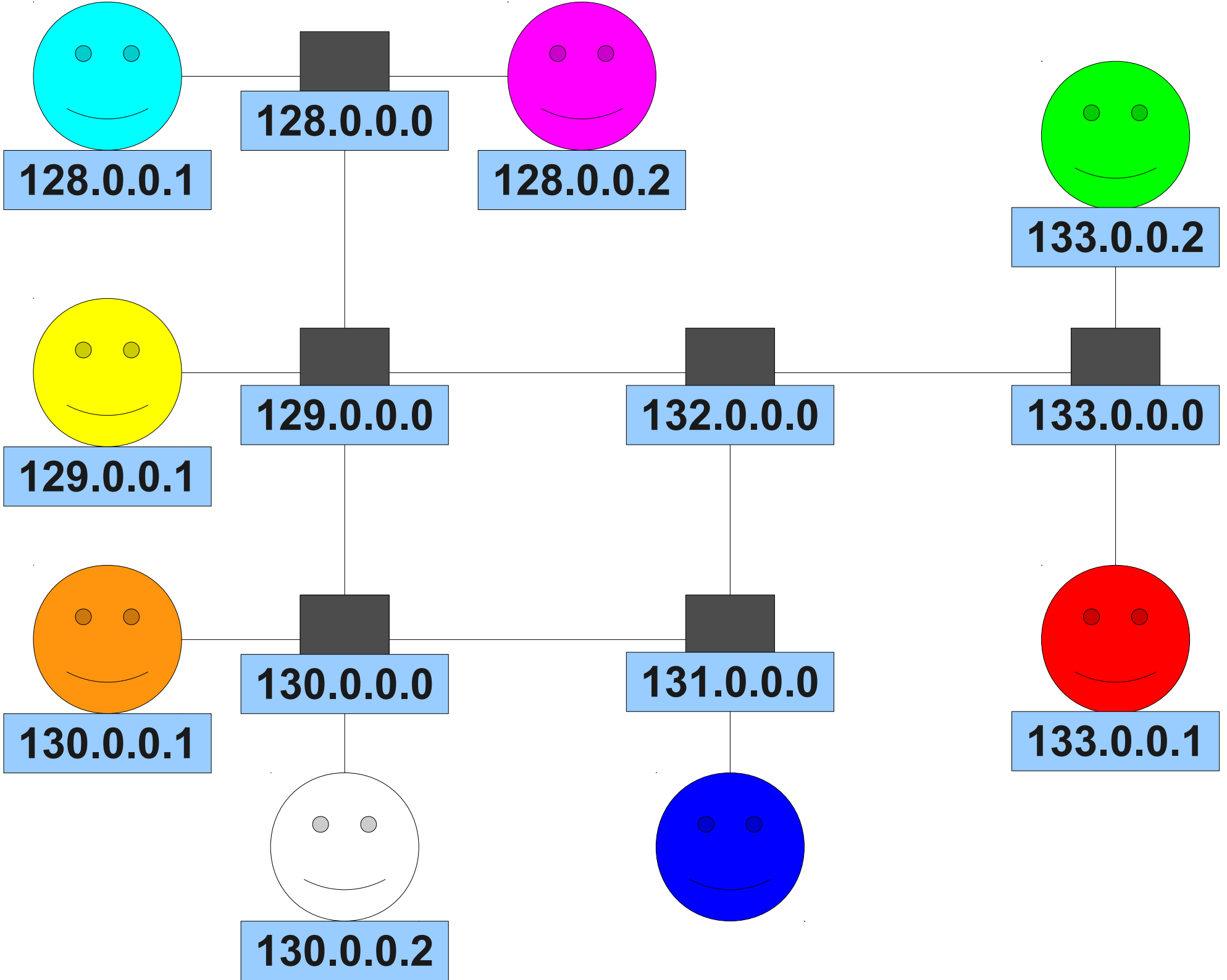

LOL

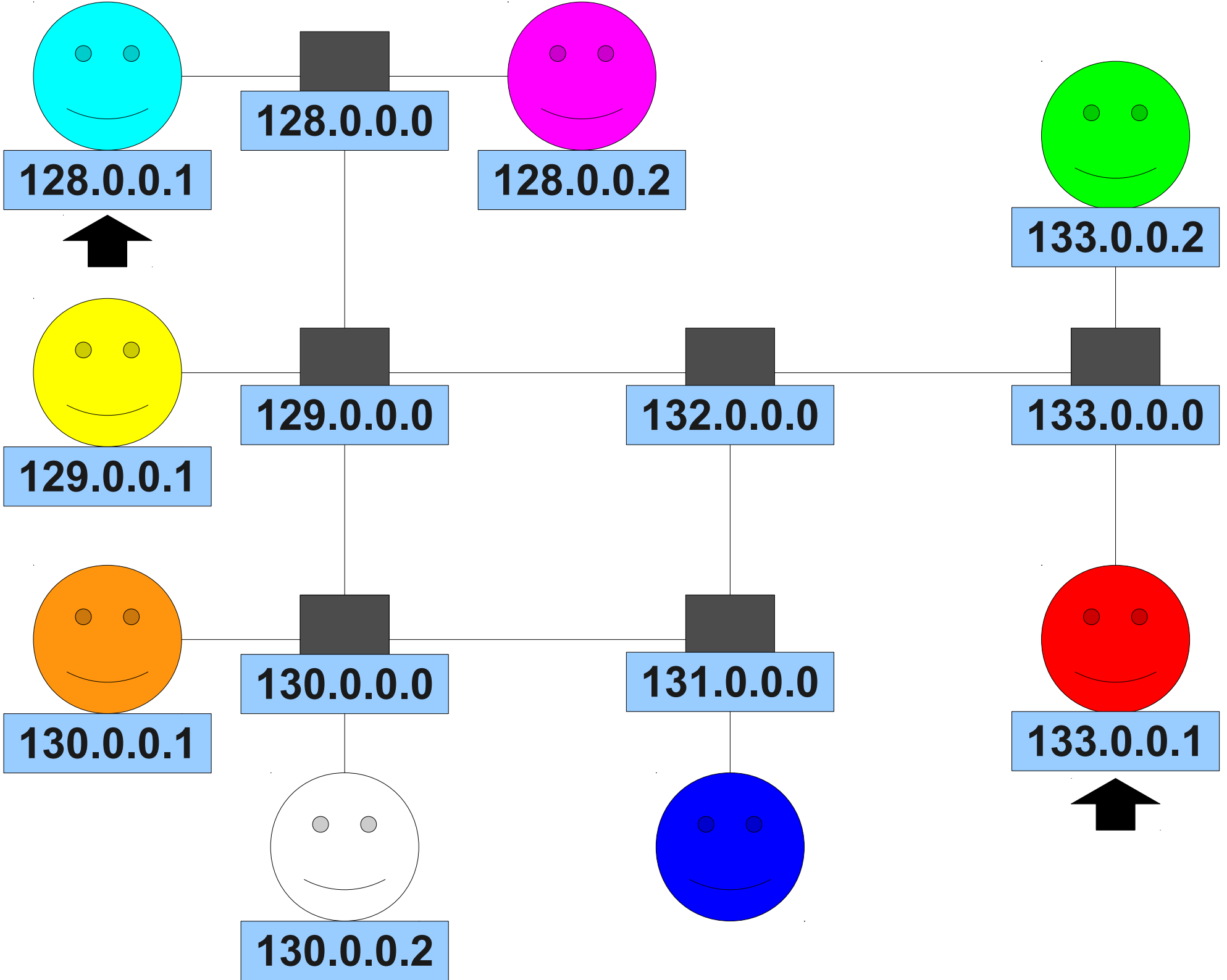

OMG
LOL

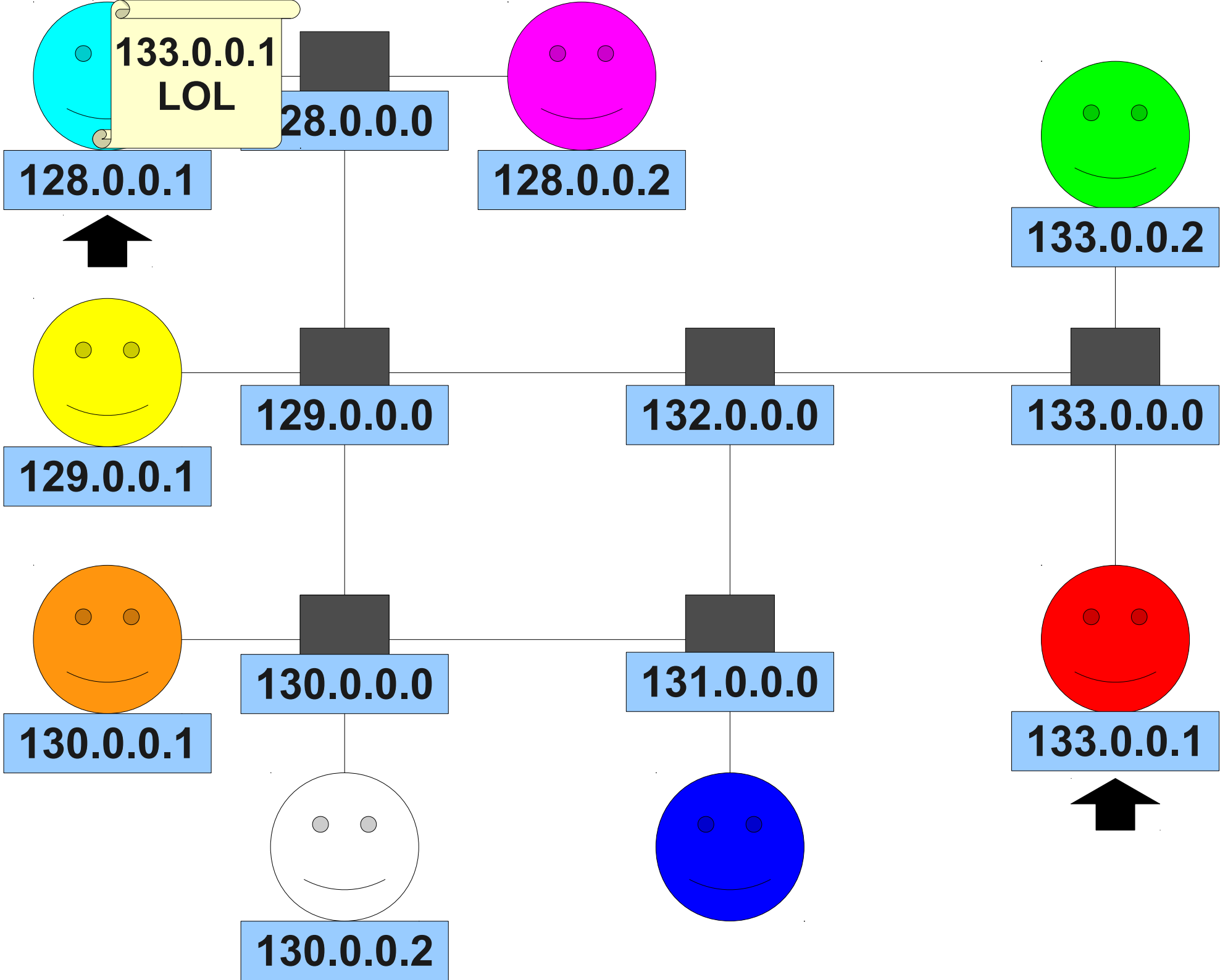
Sending Data

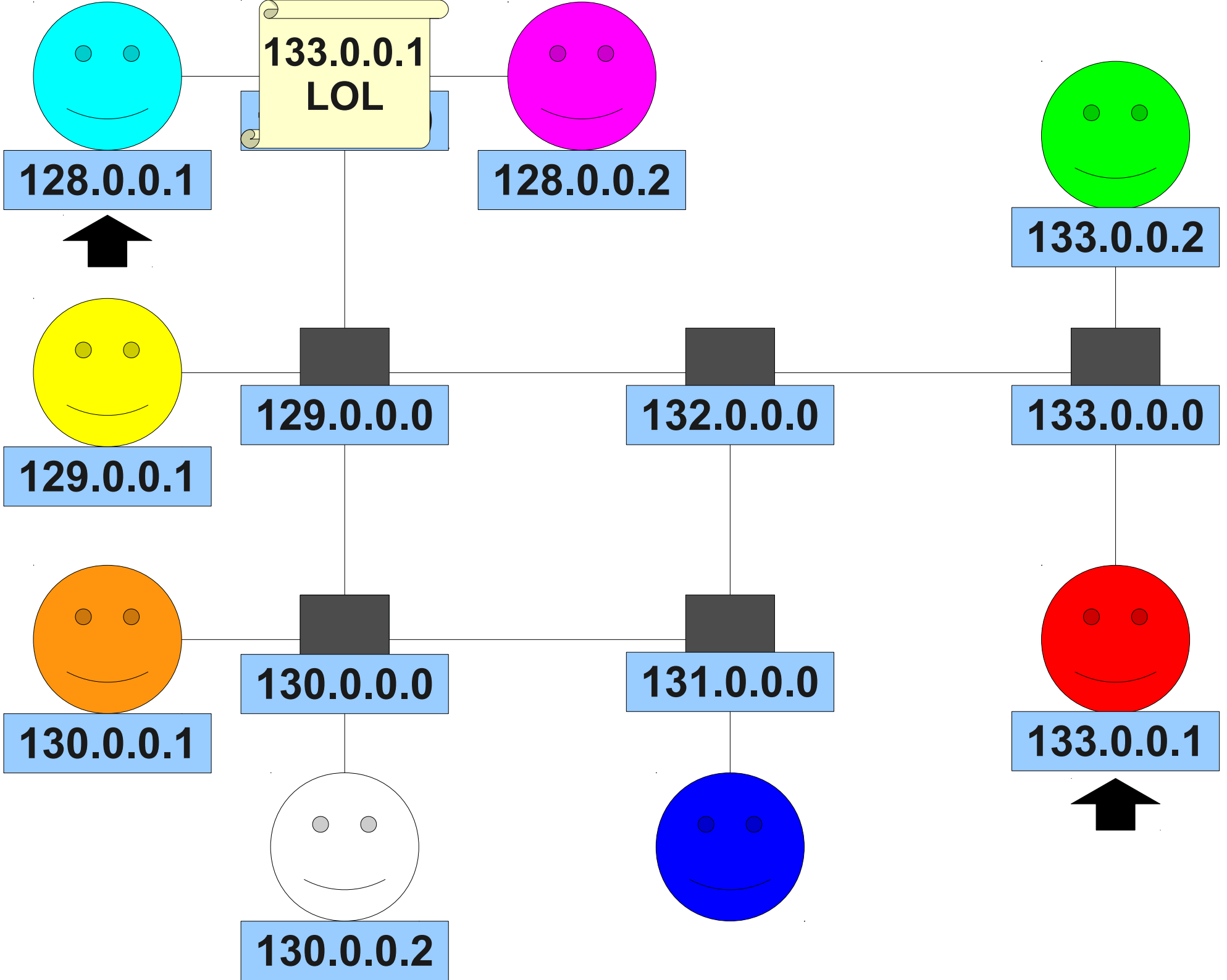
- Data is sent across the Internet in **packets**.
- Each packet contains a message (called the **payload**), along with extra information to help it get to its destination correctly.

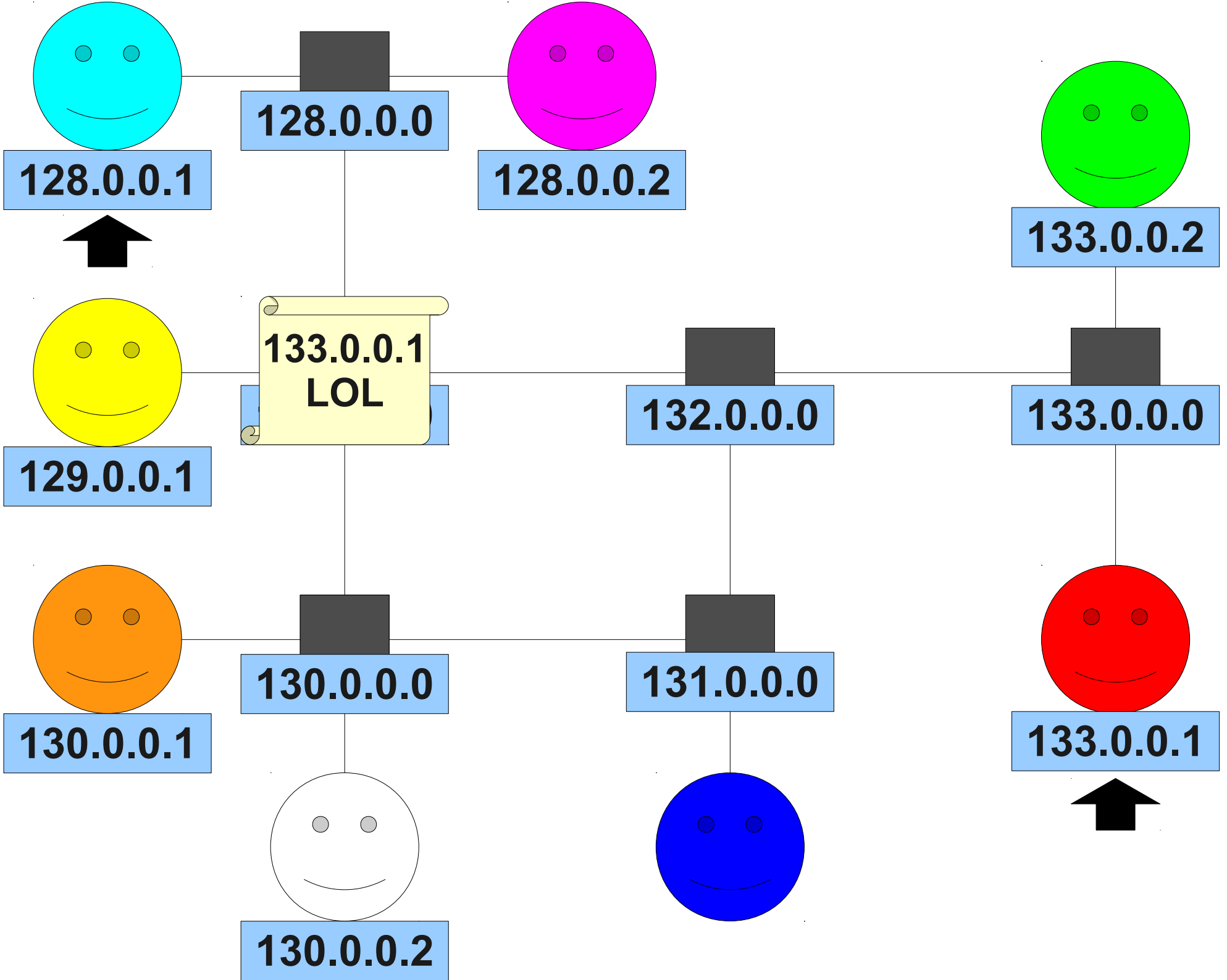


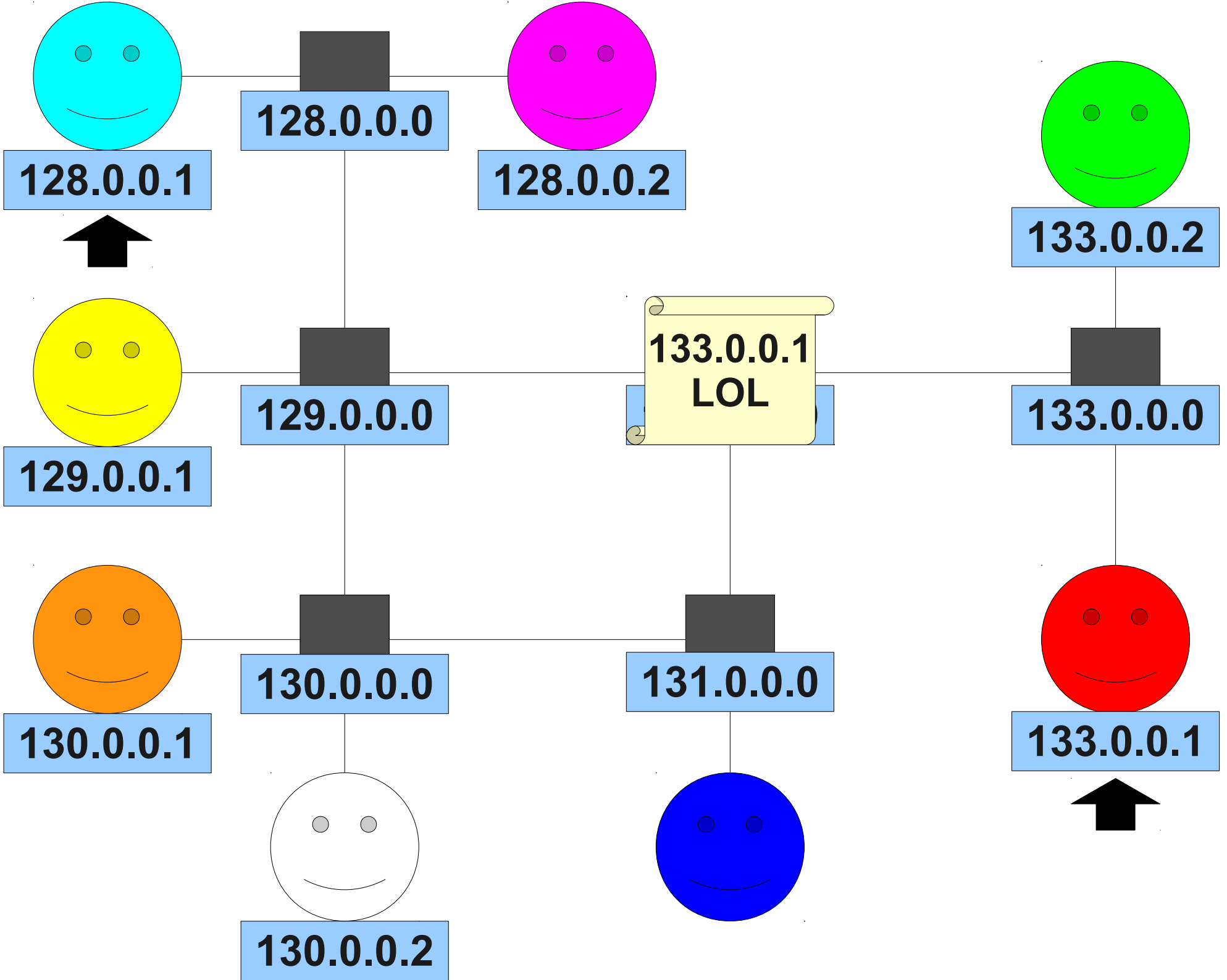

128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
132.0.0.0
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2

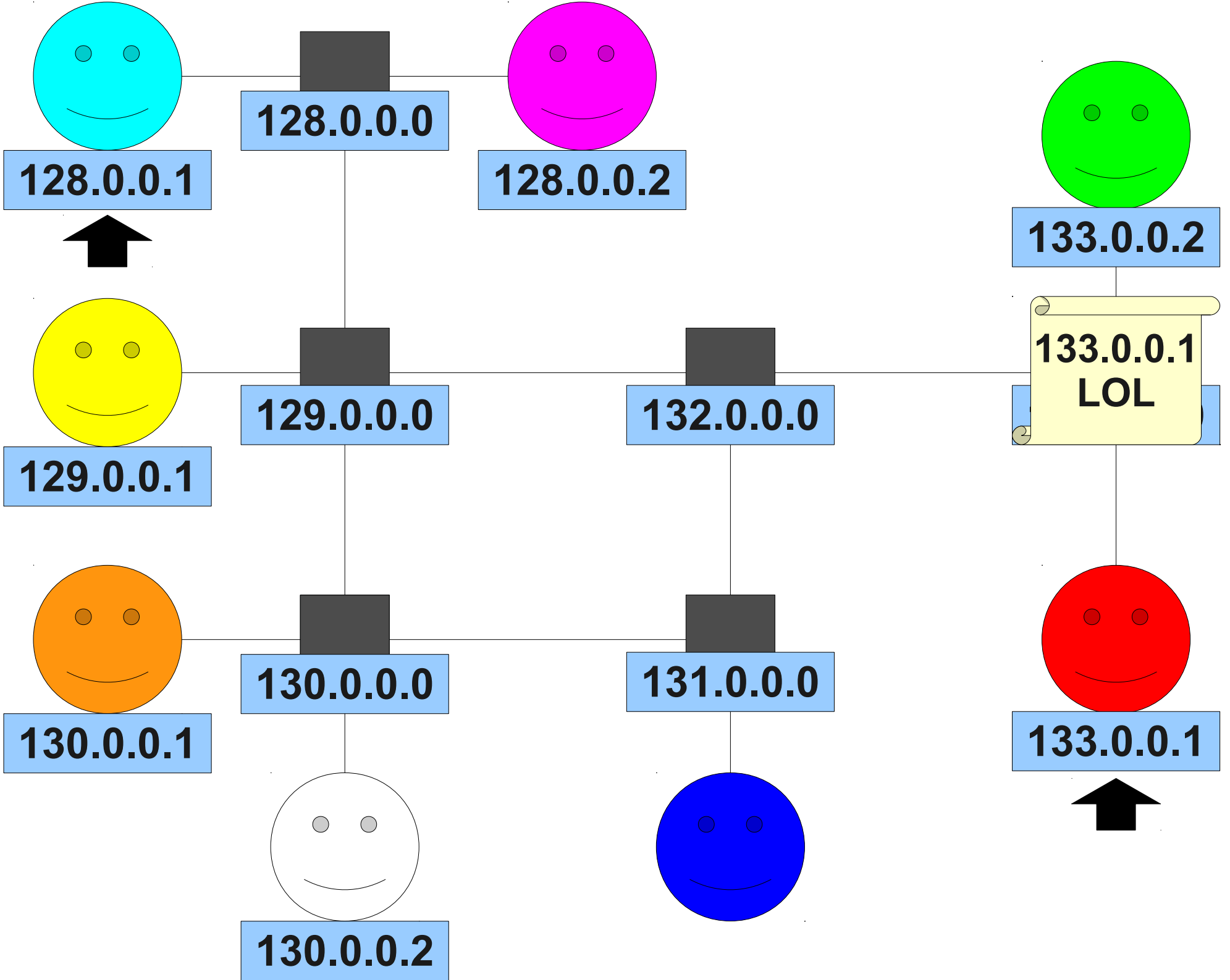

128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
132.0.0.0
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2

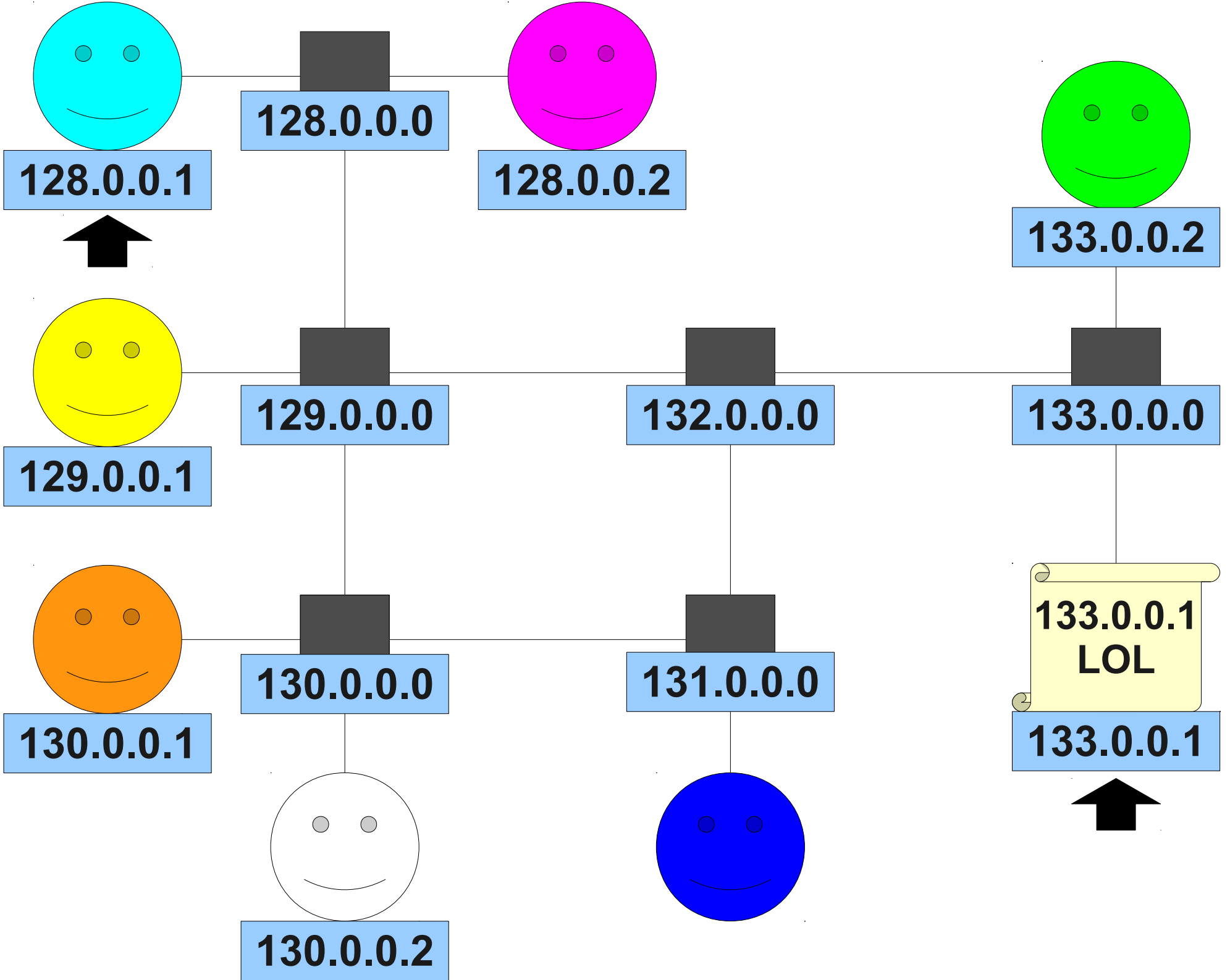

133.0.0.1
LOL
128.0.0.0
128.0.0.1
128.0.0.2
133.0.0.2
129.0.0.0
132.0.0.0
133.0.0.0
129.0.0.1
130.0.0.0
131.0.0.0
130.0.0.1
133.0.0.1
130.0.0.2

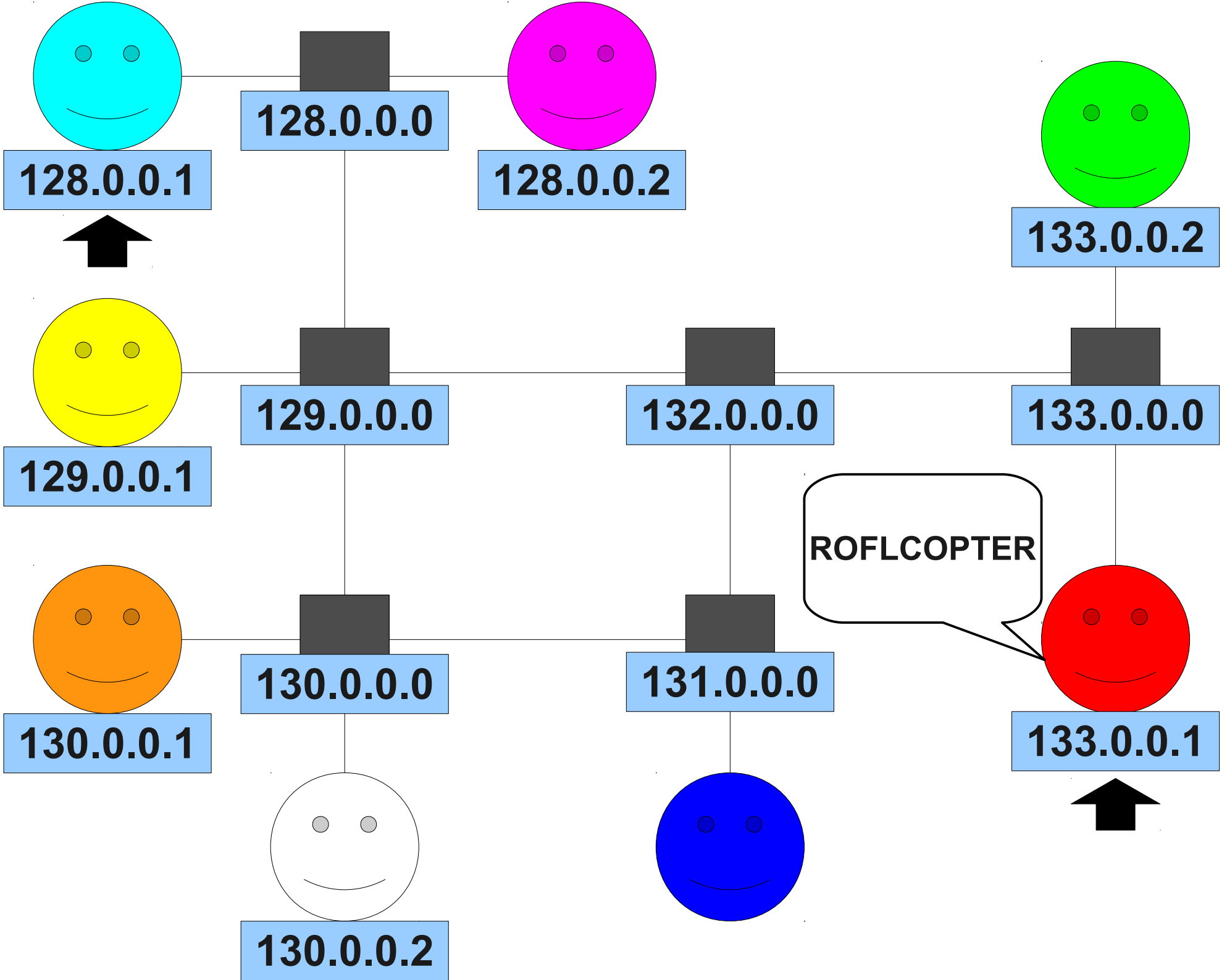

133.0.0.1
LOL
128.0.0.1
128.0.0.2
133.0.0.2
129.0.0.0
132.0.0.0
133.0.0.0
129.0.0.1
130.0.0.0
131.0.0.0
130.0.0.1
133.0.0.1
130.0.0.2


128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
133.0.0.1
LOL
132.0.0.0
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2


128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
133.0.0.1
LOL
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2


128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
132.0.0.0
133.0.0.1
LOL
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2


128.0.0.1
128.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
132.0.0.0
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
LOL
130.0.0.2


128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
132.0.0.0
133.0.0.0
ROFLCOPTER
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2

IP Addresses

- Each computer may have one or more **IP addresses** so that it can receive messages over the Internet.
  - Similar to a phone number.
- There are two types of IP addresses:
  - IPv4: 2^{32} possible addresses (about four billion), and we're rapidly running out!
  - IPv6: 2^{128} possible addresses (about 4×10^{34}), and we're very unlikely to run out in the future.

Hostnames

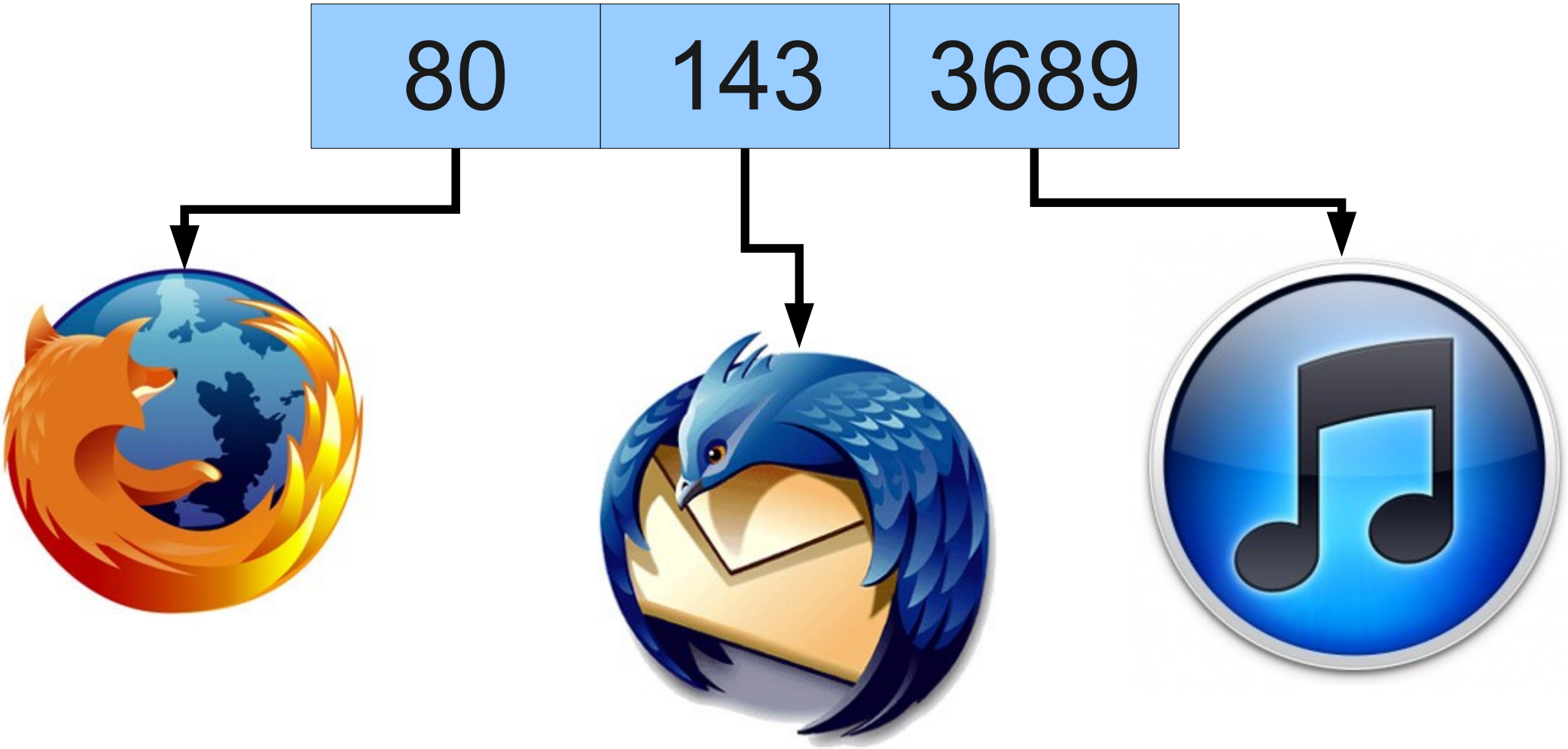
- In order to make it easier to find remote computers, computers can have names associated with them.
  - www.google.com
  - www.stanford.edu
- These names are called **hostnames**.
- A system called the **domain name system** is responsible for converting domain names into IP addresses.
  - Kind of like a huge `Map<String, IPAddress>`

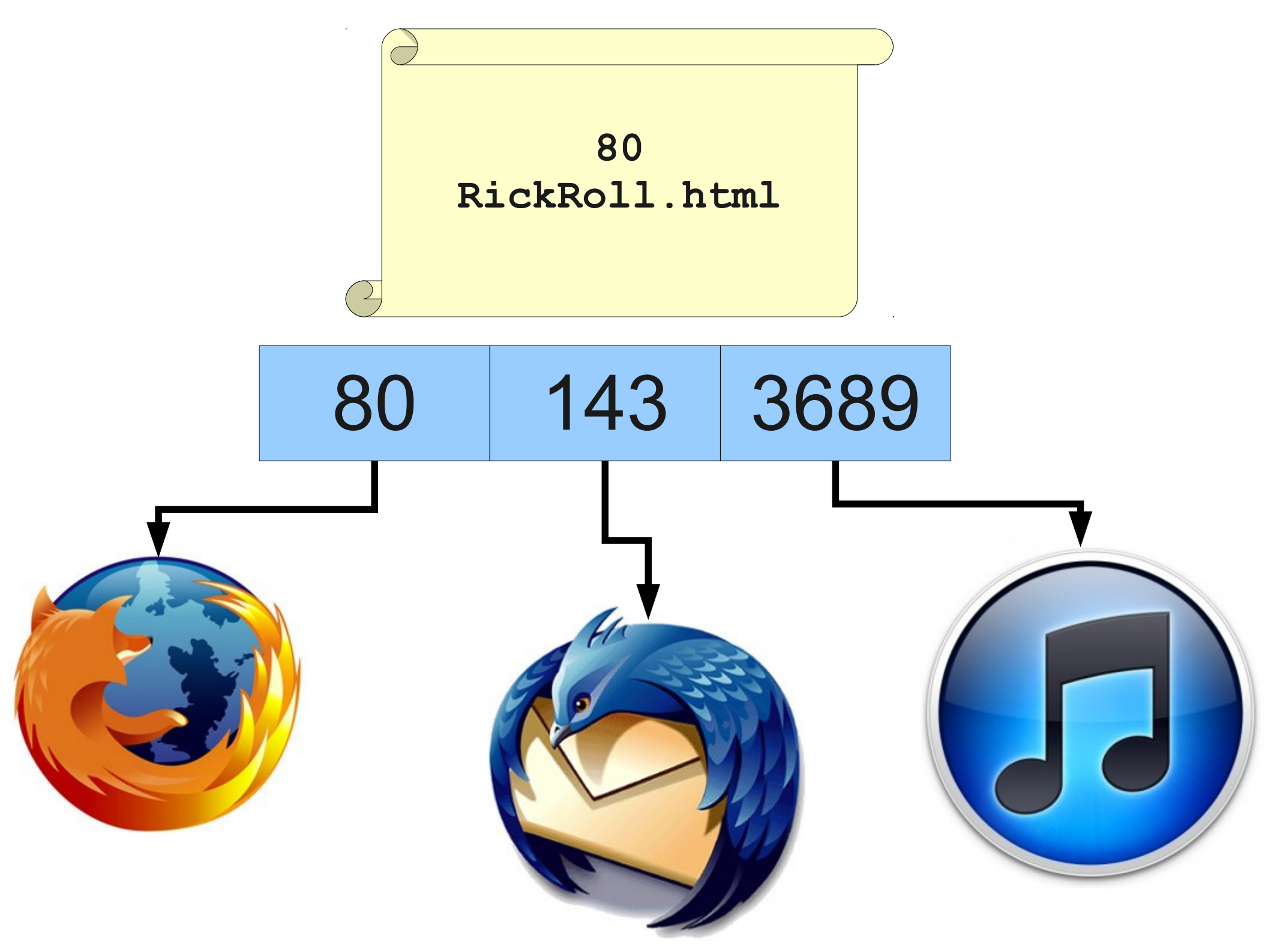
A Small Problem

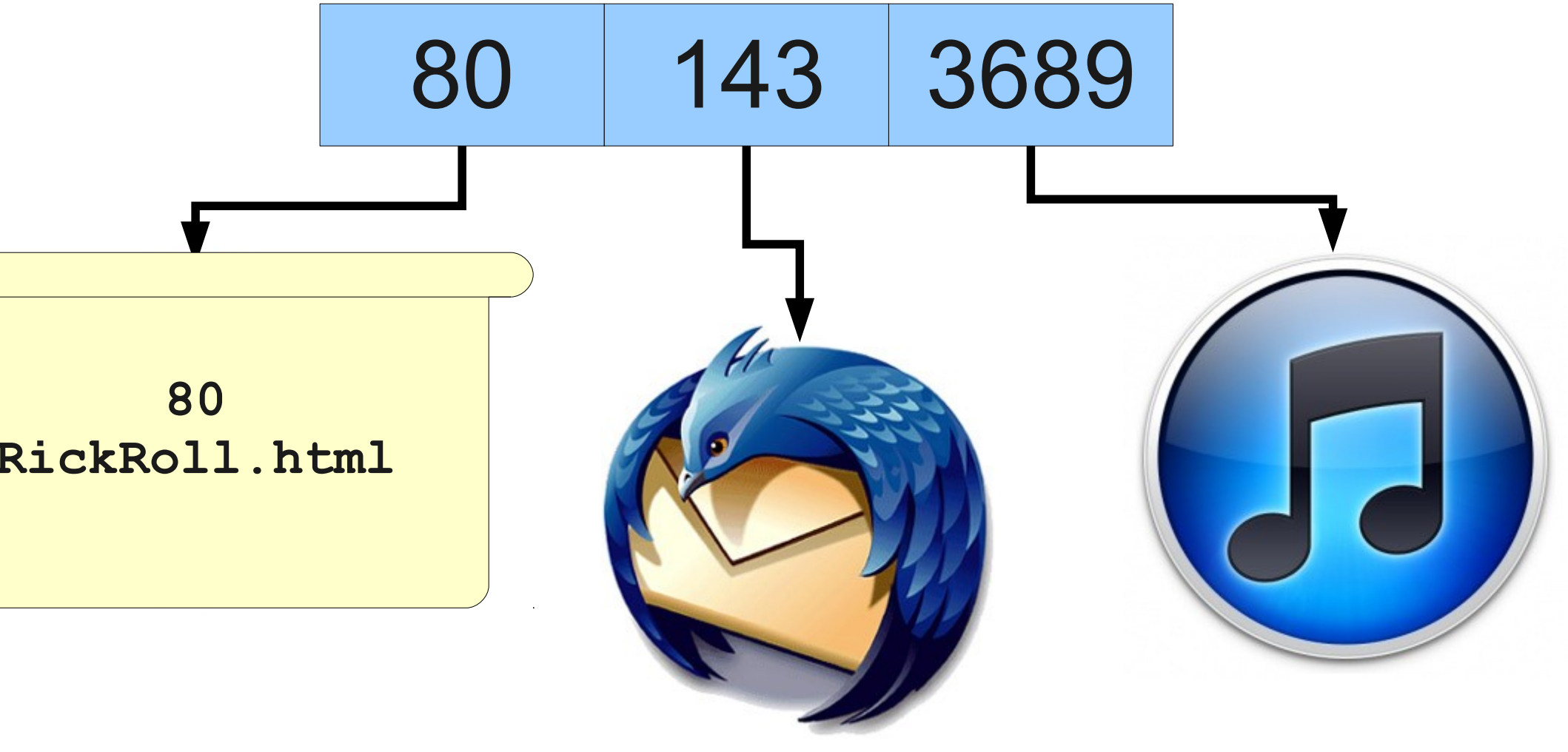
- At any one time, you could be
  - Surfing the web,
  - Downloading music from iTunes,
  - Checking your email,
  - Chatting on IM,
  - etc.
- You might have packets from many different machines all arriving at once.
- How does the computer know how to send each message to the right program?

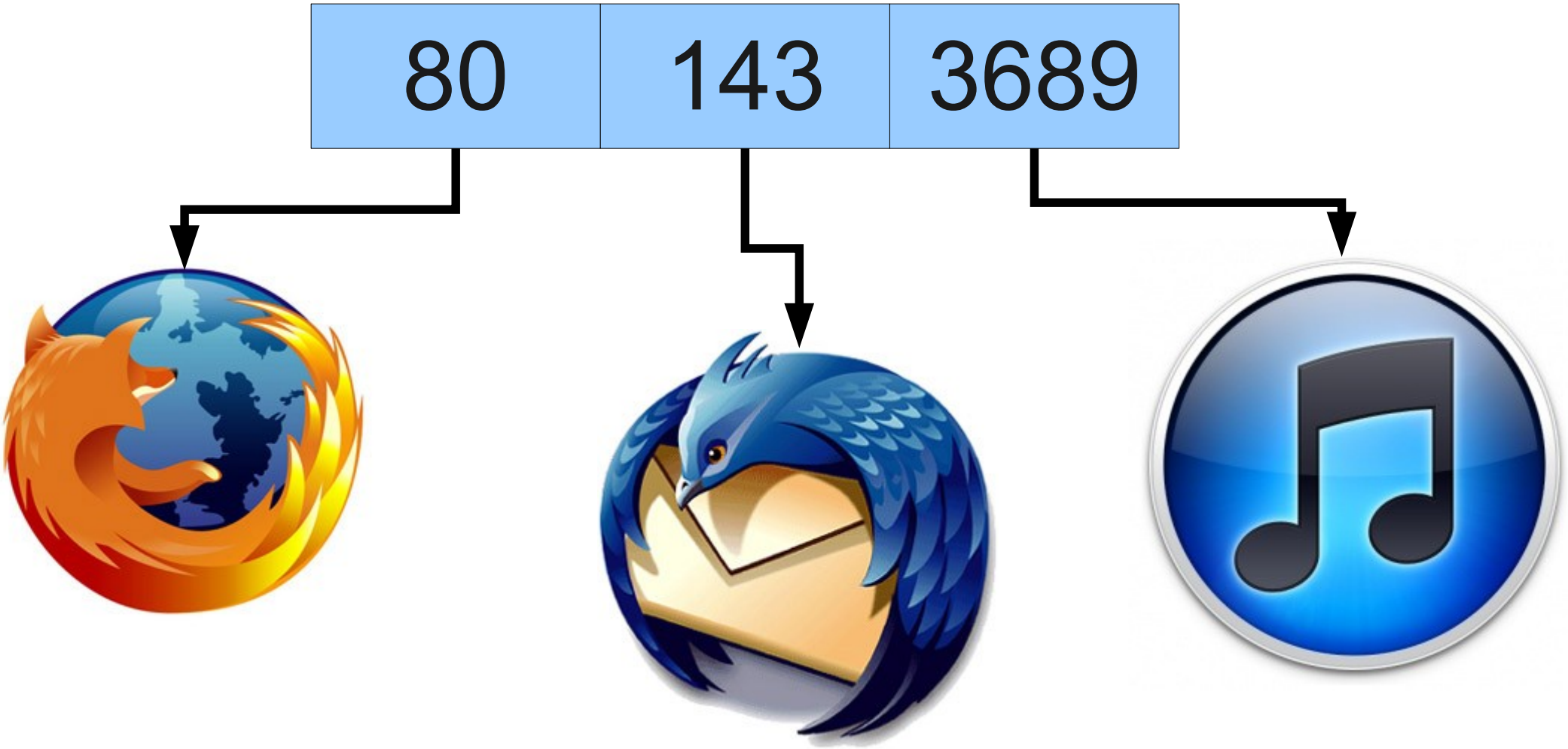
Ports

- Every packet is labeled with a **port number** that lets the destination computer know how to process the message.
- Different applications listen in on different ports:
  - Sending mail (SMTP): Port 25
  - Browsing the web (HTTP): Port 80
  - Checking email (IMAP): Port 143


80
143
3689

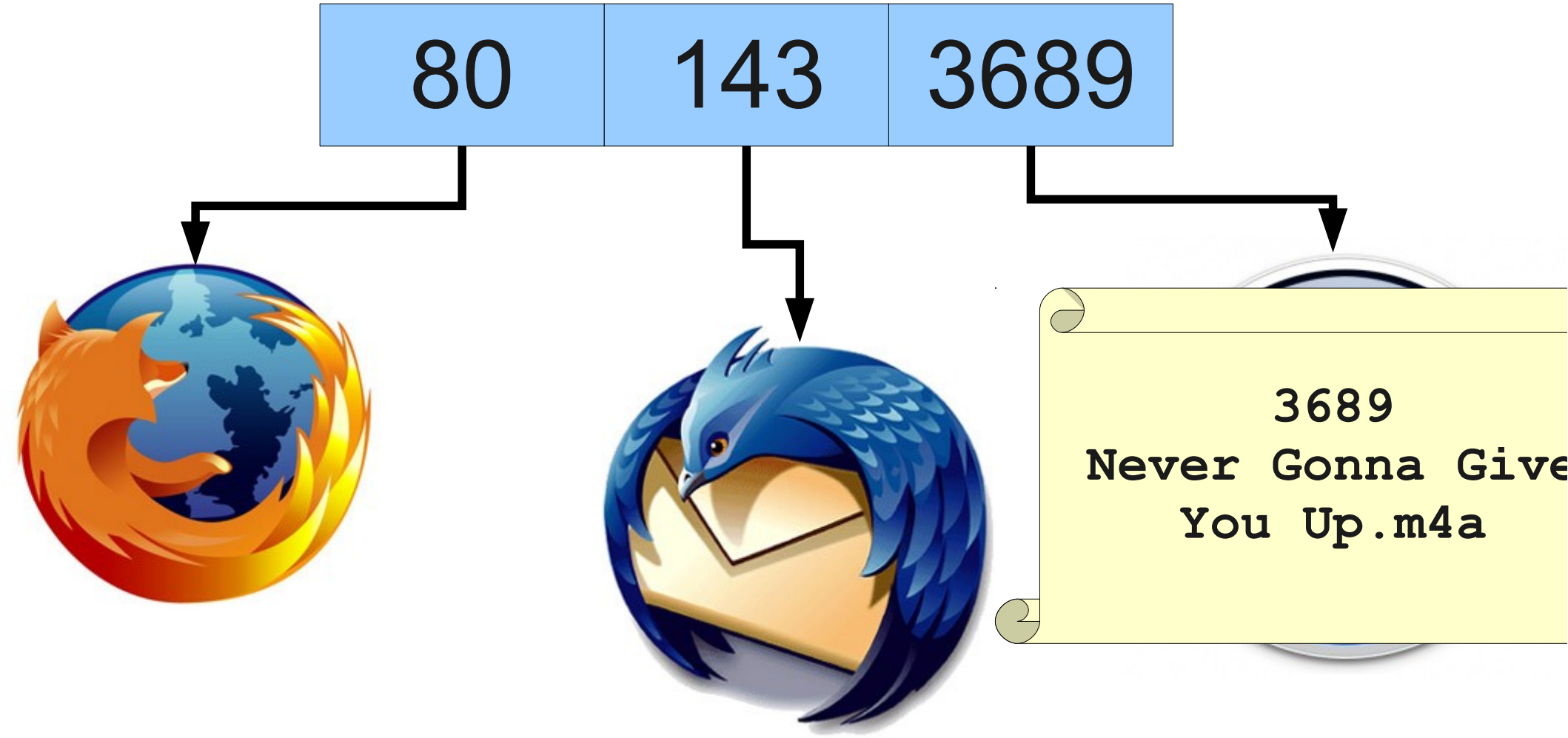

80
RickRoll.html
80
143
3689

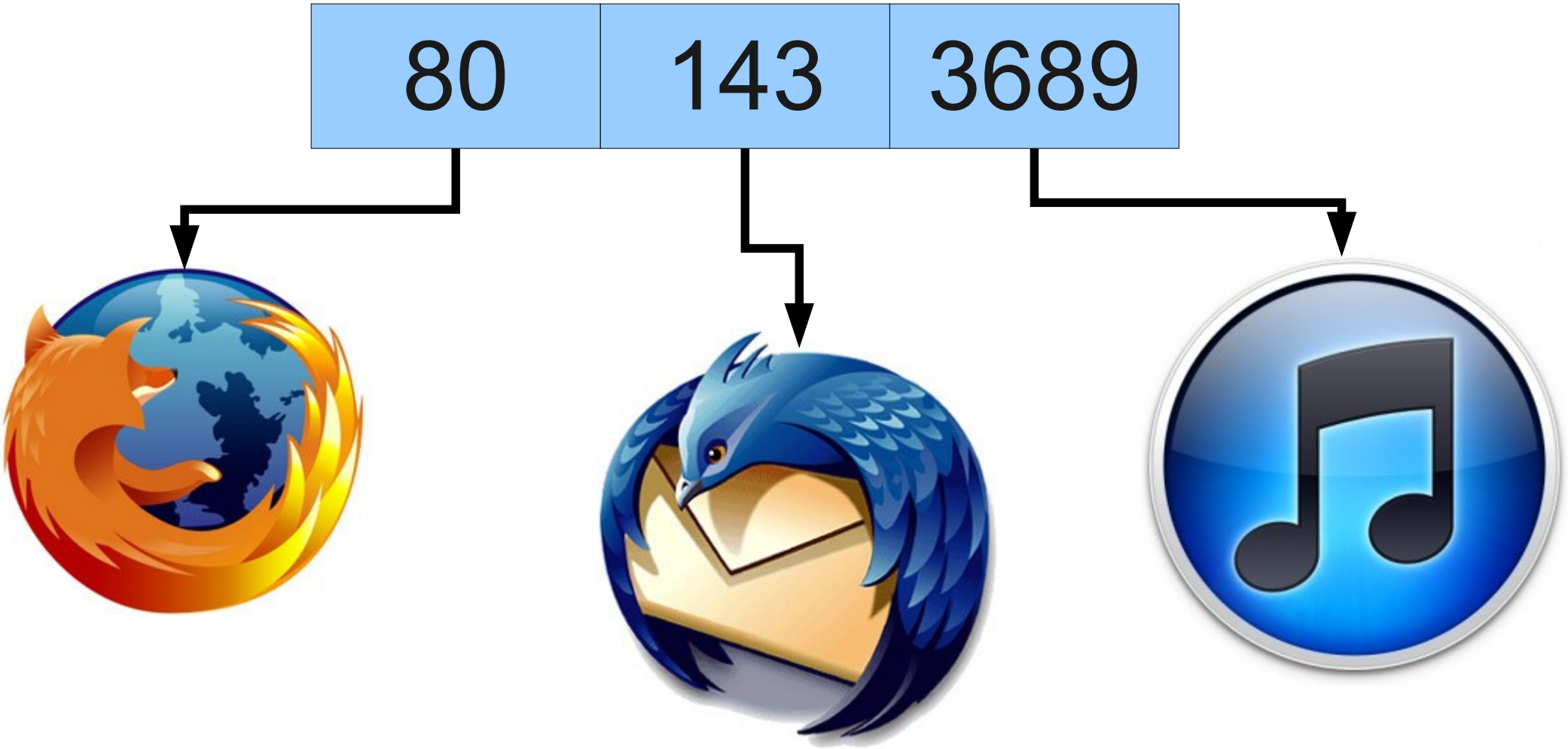

80
143
3689
80
RickRoll.html


80
143
3689

3689
Never Gonna Give You Up.m4a

80	143	3689


80
143
3689
3689
Never Gonna Give
You Up.m4a


80
143
3689

Sockets

- A **socket** is a combination of an IP address (destination computer) and port number (what program should read the message).
- All the information necessary to ensure that a message gets to the right program on the right computer.
- To set up a connection to a remote computer, you need to create a socket connection to that computer.

Application Protocols

- Now that we can get computers talking to one another, how do they communicate information in a meaningful way?
- An **application protocol** is a set of rules computers can follow to communicate over a network.
- Each computer follows the rules of the protocol to share information.

An Example: **HTTP**

Networking in Java

- To connect to a remote machine:
  - Create a socket connection to the machine by giving a combination of the host name and the port.
  - Create a **`BufferedReader`** to read messages coming from the other computer.
  - Create a **`PrintWriter`** to send messages to the other computer.
  - Send and receive messages as you see fit!

Client/Server Architecture

- A **server** is a program that waits for incoming connections.
  - Typically, has some data or service that it can provide.
- A **client** is a program that initiates a connection to a server.
  - Typically, wants to use that data or service.
  - The program we just wrote was a client that connected to a remote web server.

Acting as a Server

- A program can act as a server as follows:
  - Create a **`ServerSocket`** on a given port and wait for an incoming connection.
  - Obtain a **`Socket`** that lets you communicate with the machine that has connected.
  - Proceed as before.

A Simple Chat Program