Security and Privacy

Outline for Today

- **Iterators and Assignment 7**
  - One final topic for the last assignment.
  - Demo for Assignment 7.
- **Security and Privacy**
  - A case study in privacy and security.

Iterators

- To visit every element of a collection, you can use the "for each" loop:

```
for (ElemType elem: collection) {
    ...
}
```

- Alternatively, you can use an **iterator**, an object whose job is to walk over the elements of a collection.
- The iterator has two commands:
  - **`hasNext()`**, which returns whether there are any more elements to visit, and
  - **`next()`**, which returns the next element and moves the iterator to the next position.

Java Iterators

```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators


```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

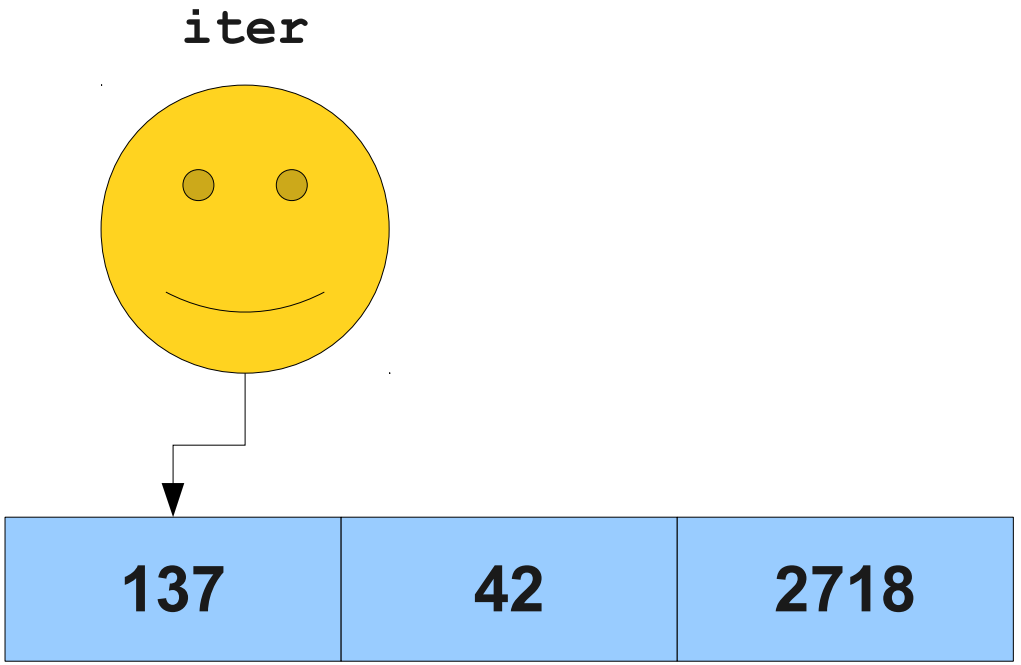
Java Iterators


```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

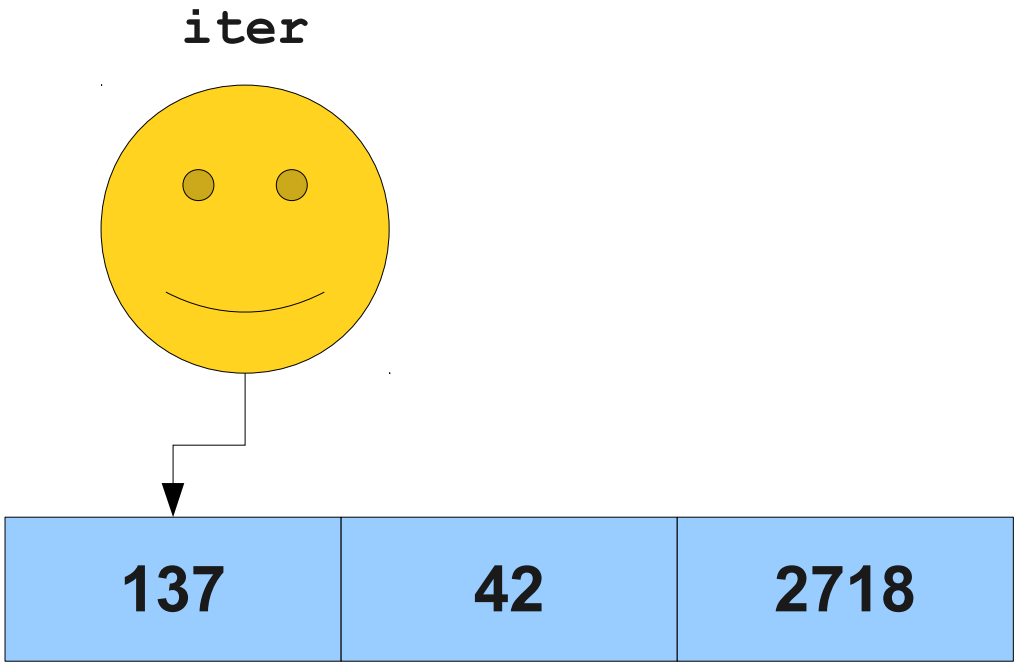



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

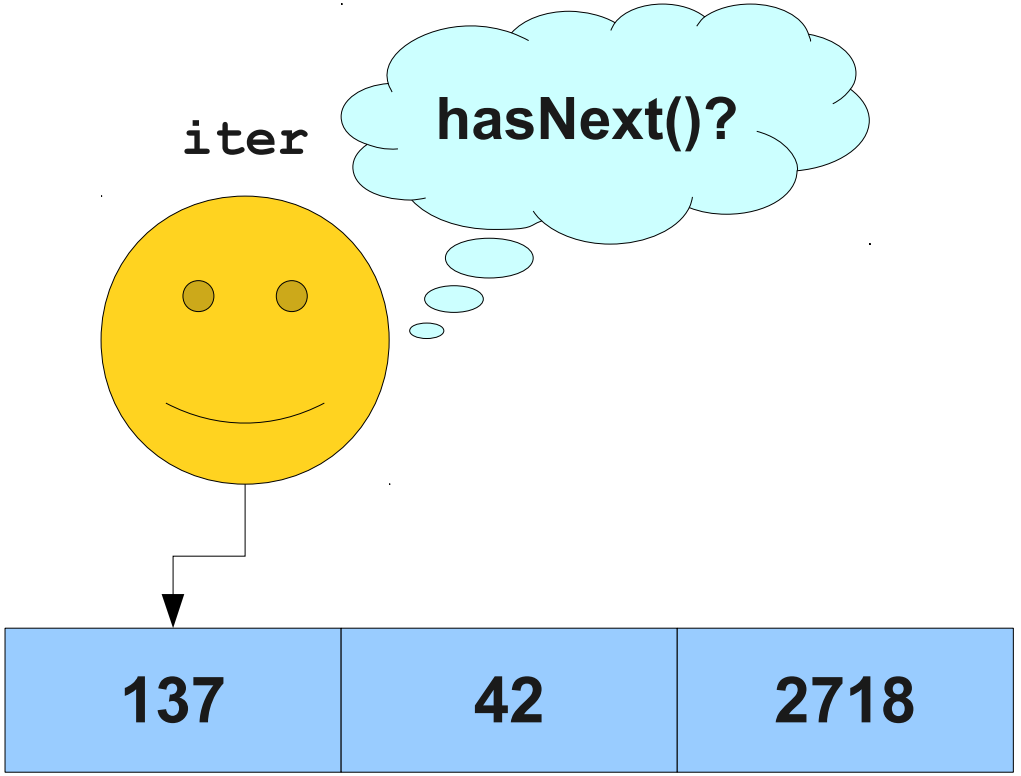



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

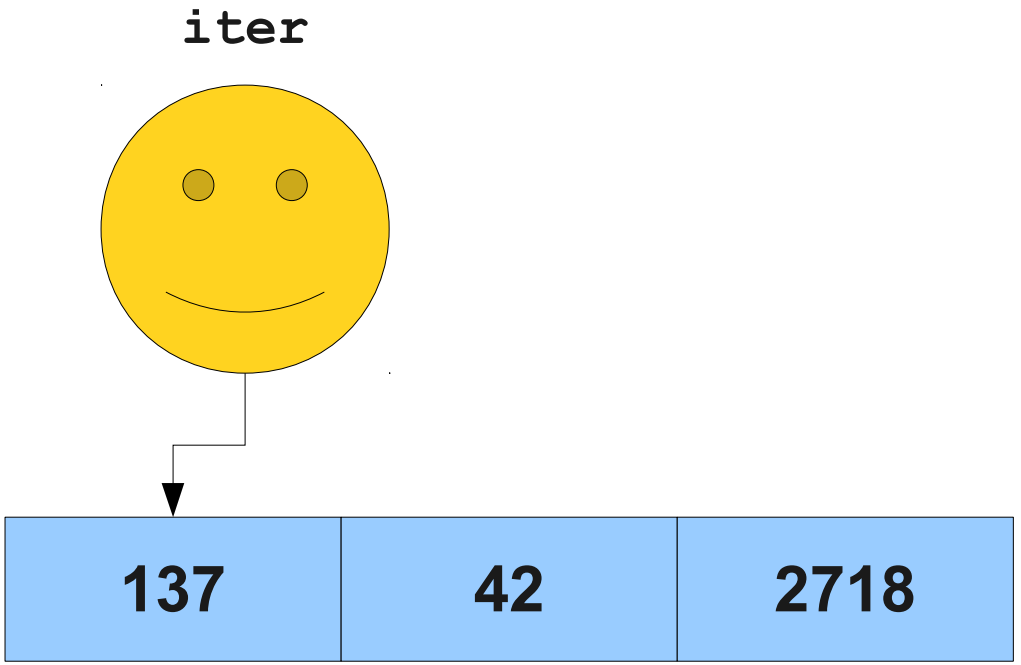



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

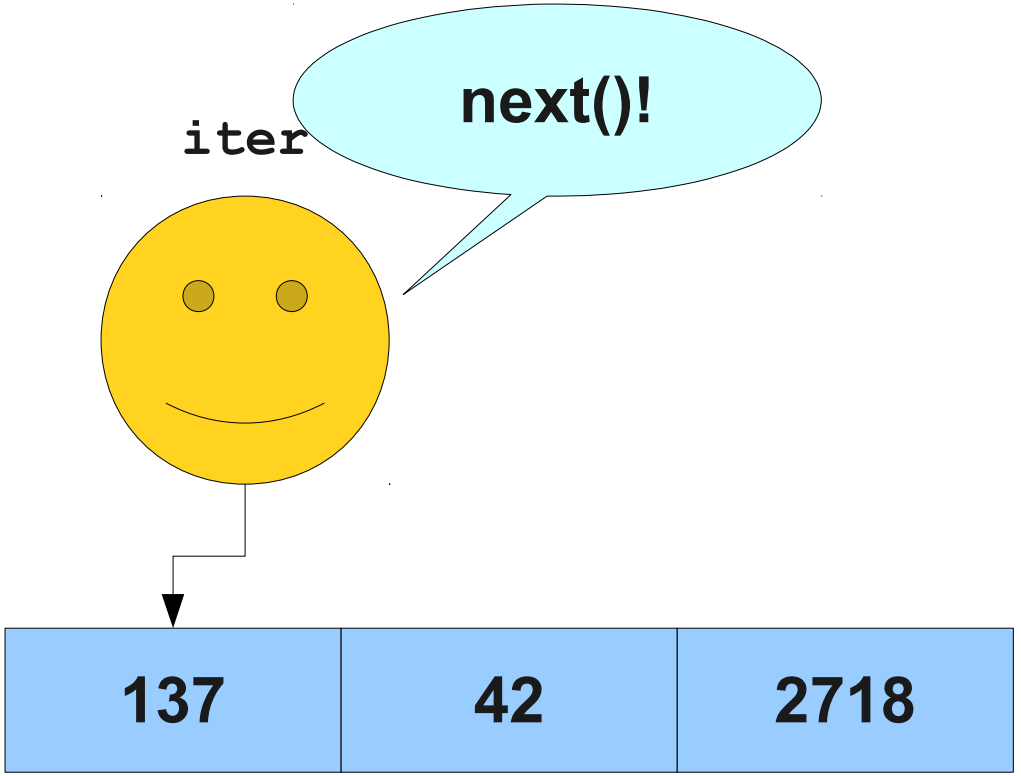



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

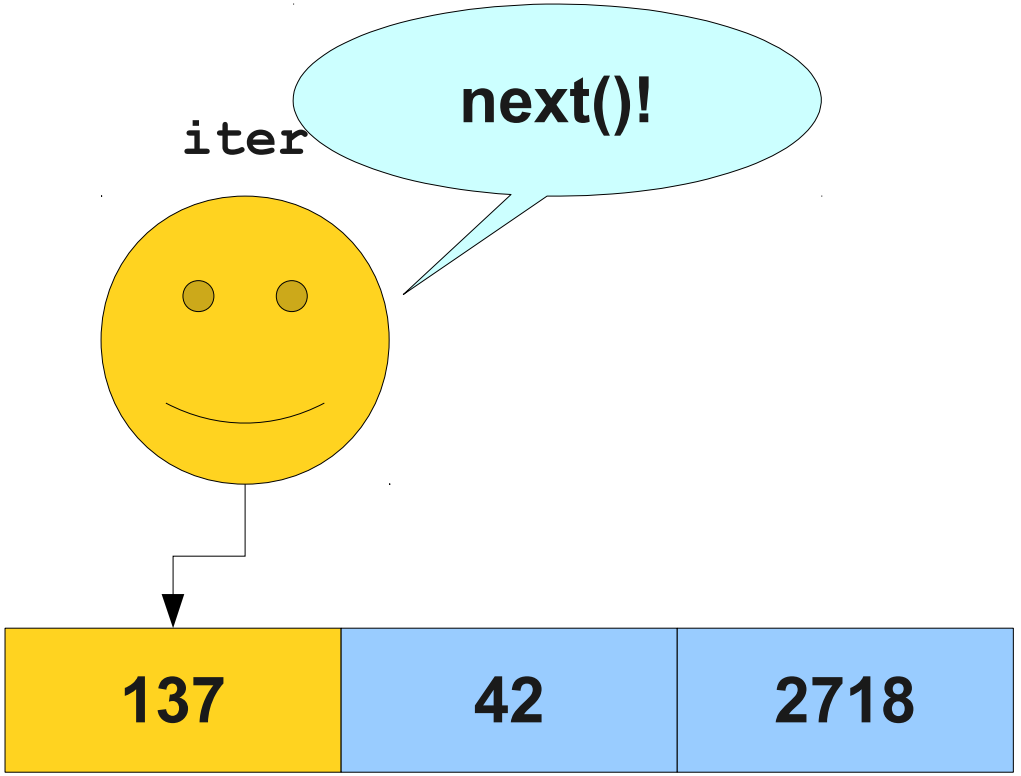



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

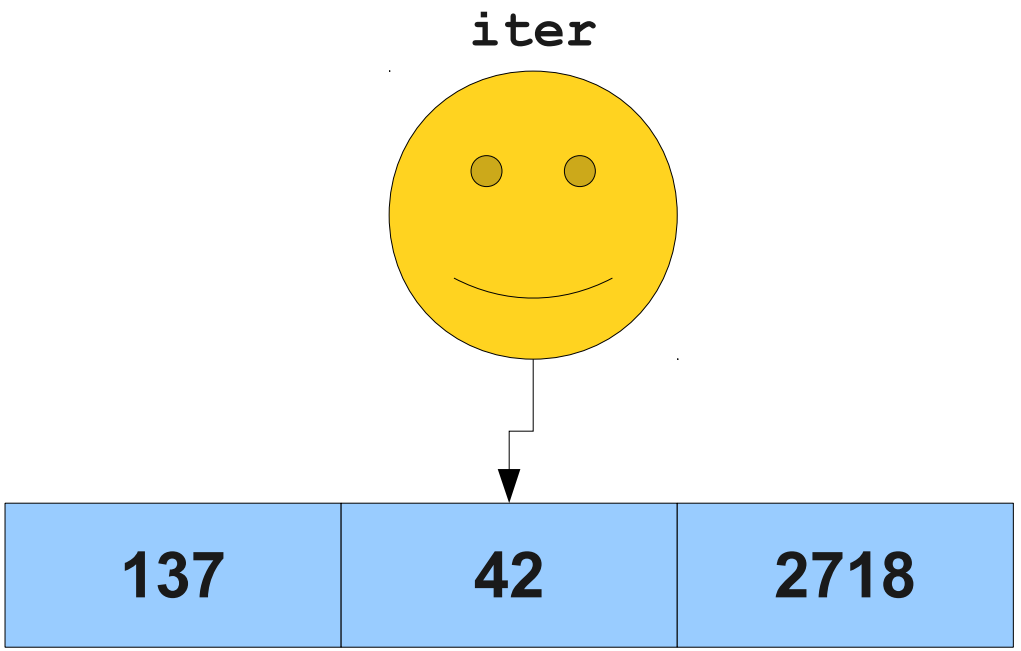



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

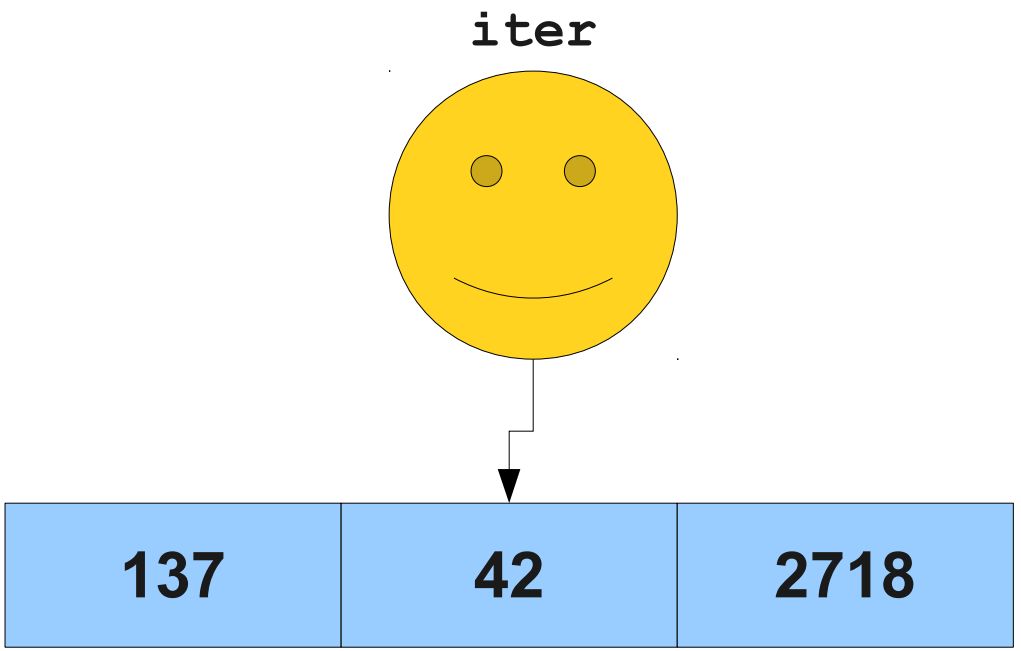



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

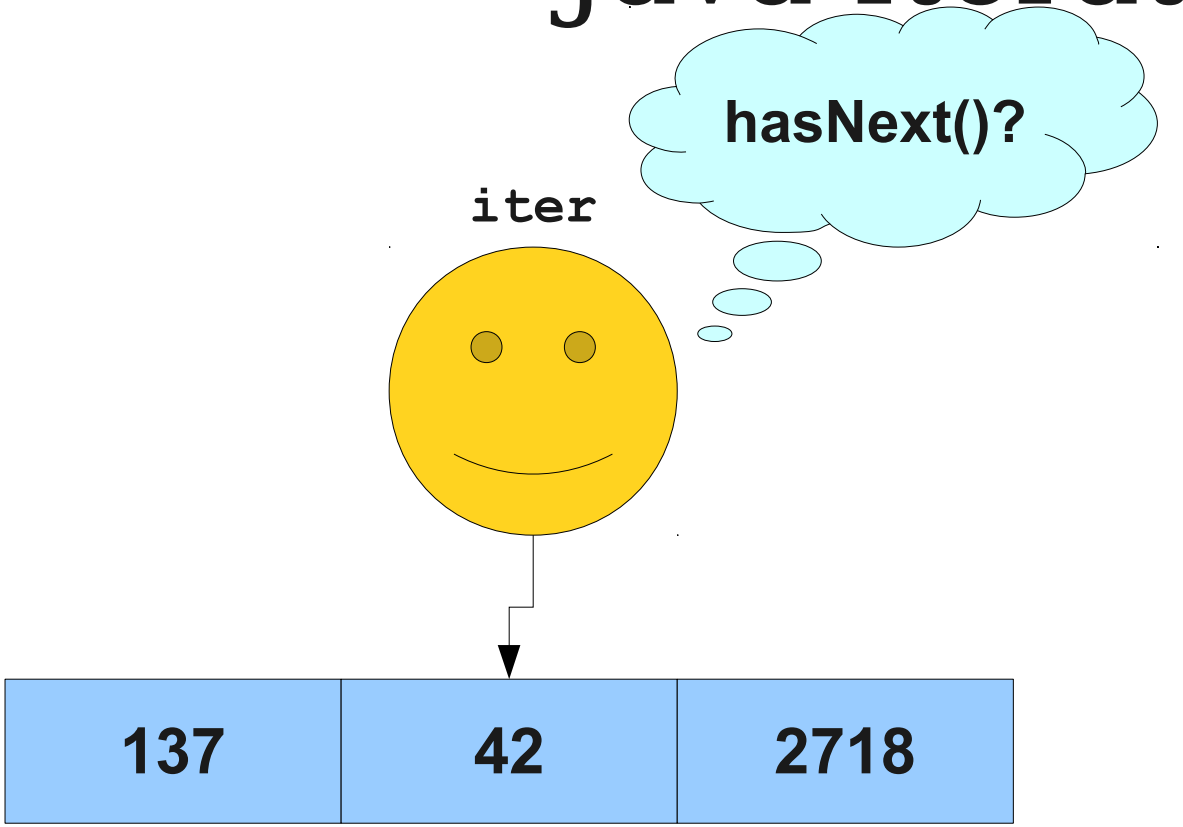



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

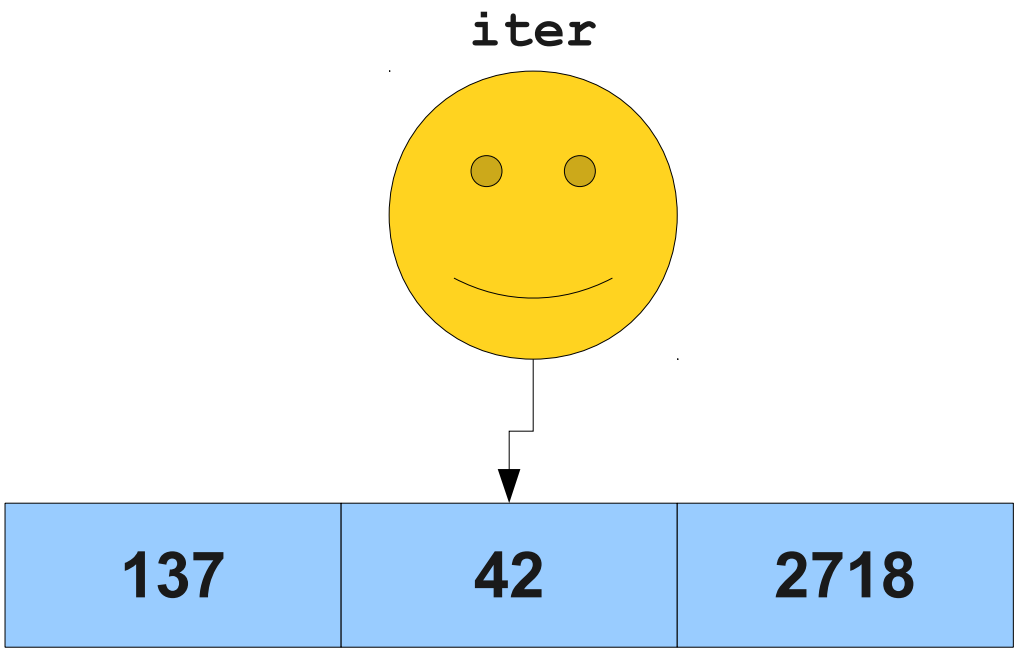



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

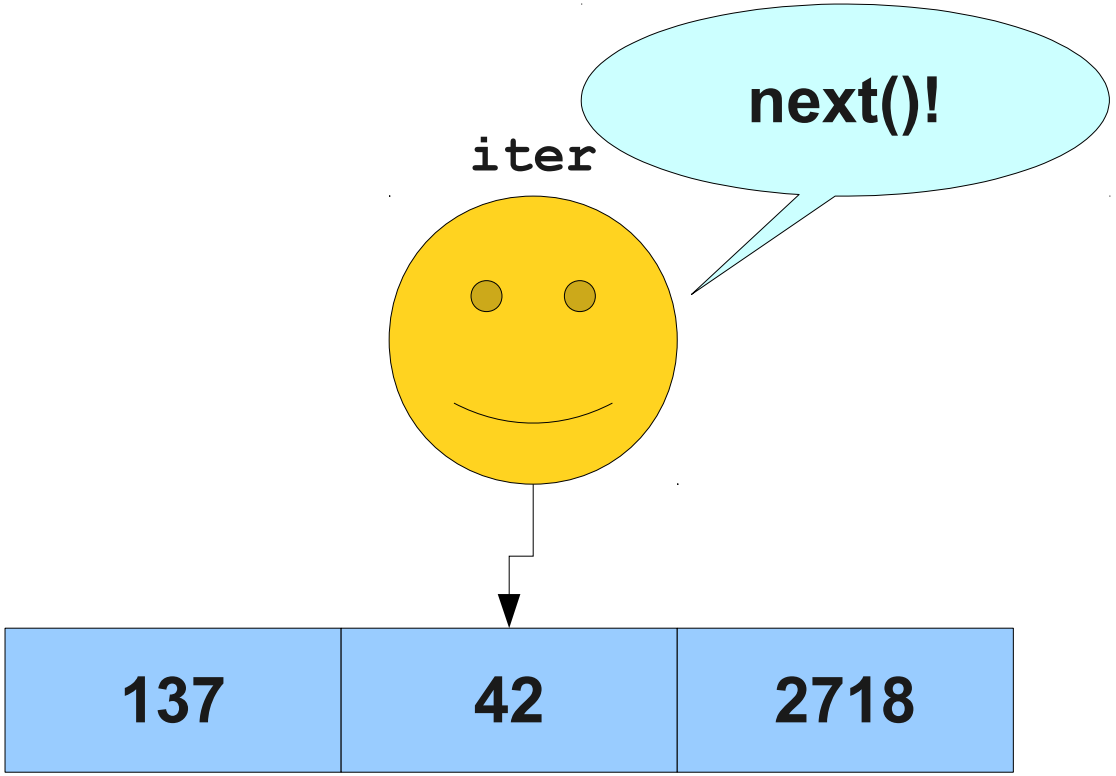



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

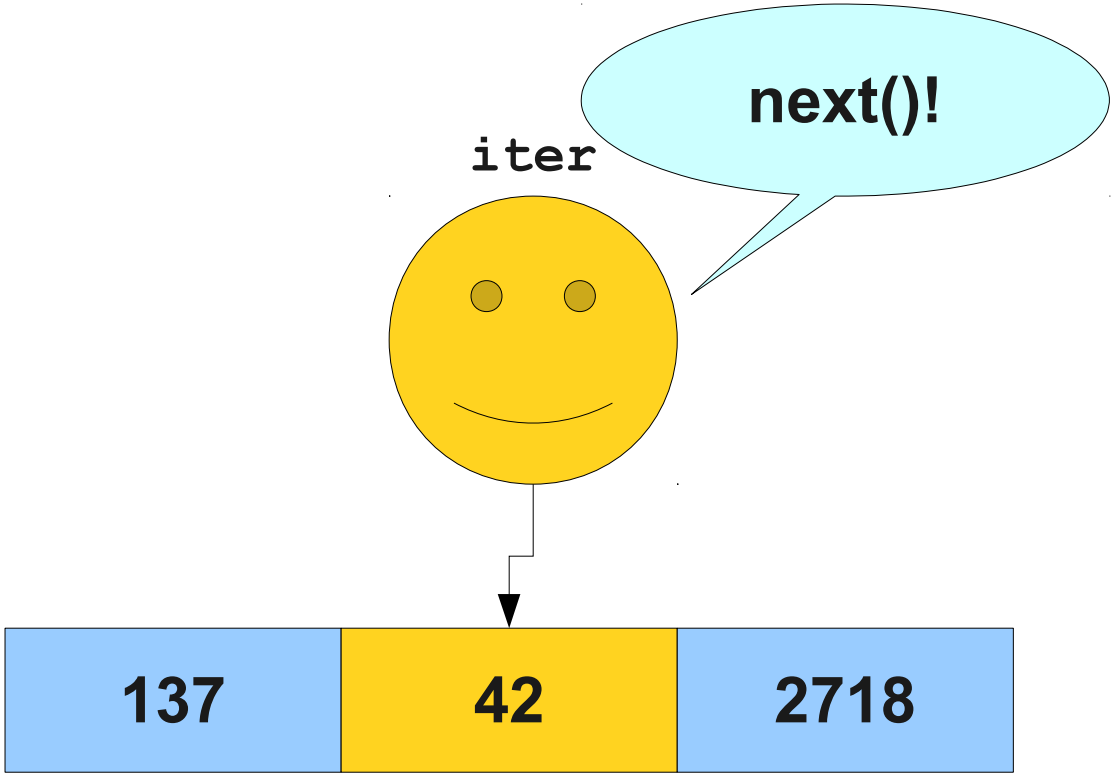



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

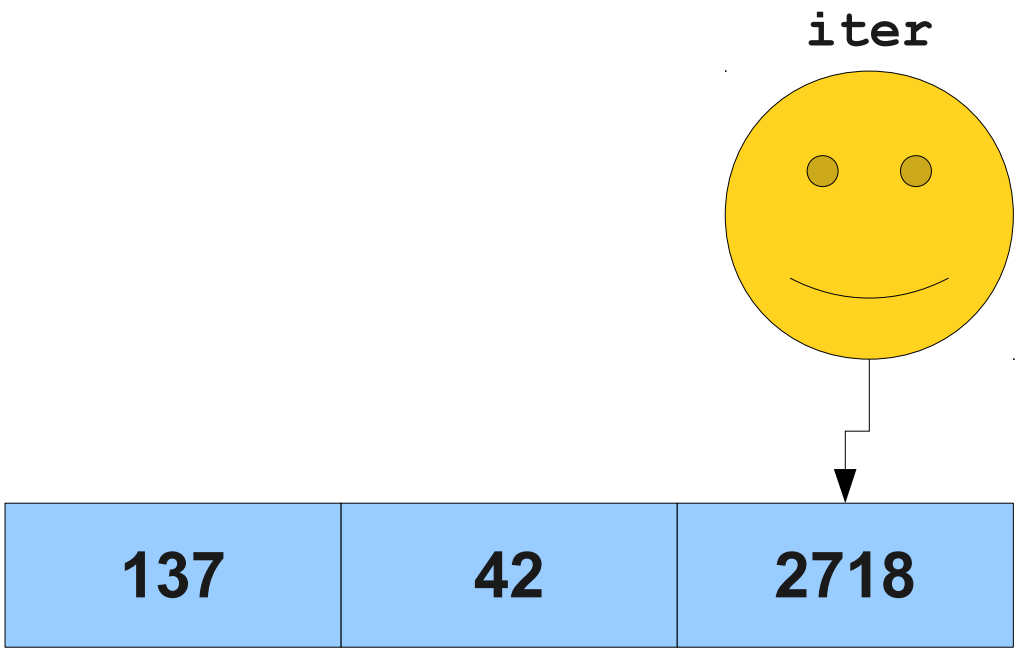



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

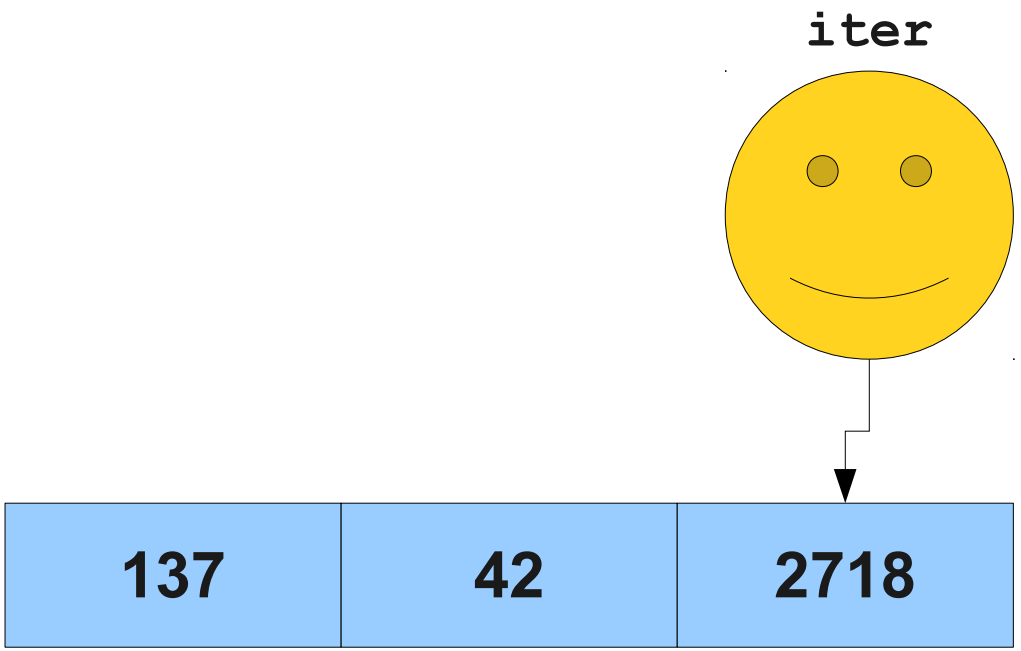



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

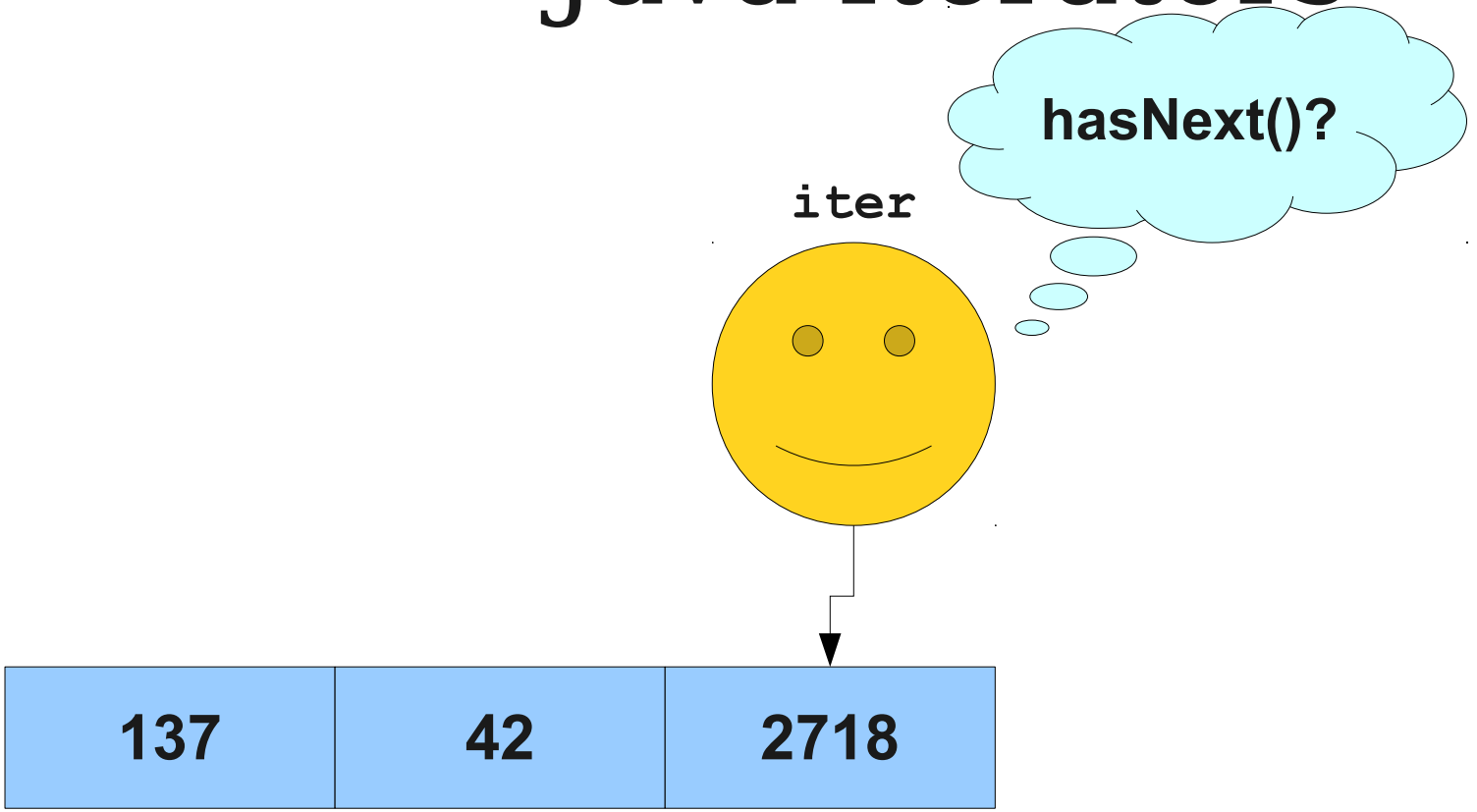



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

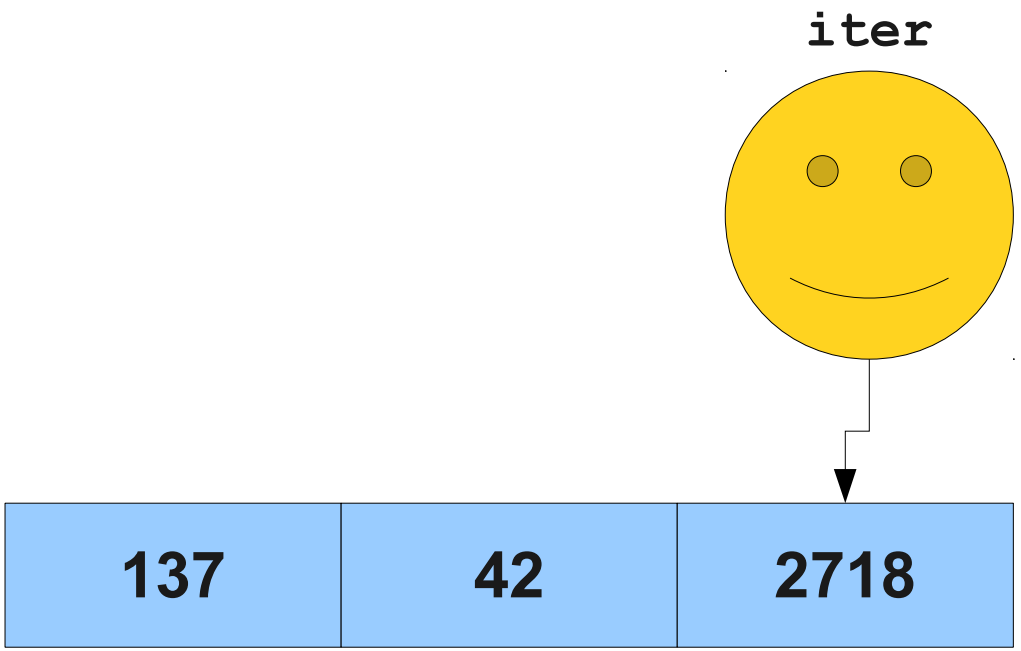



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

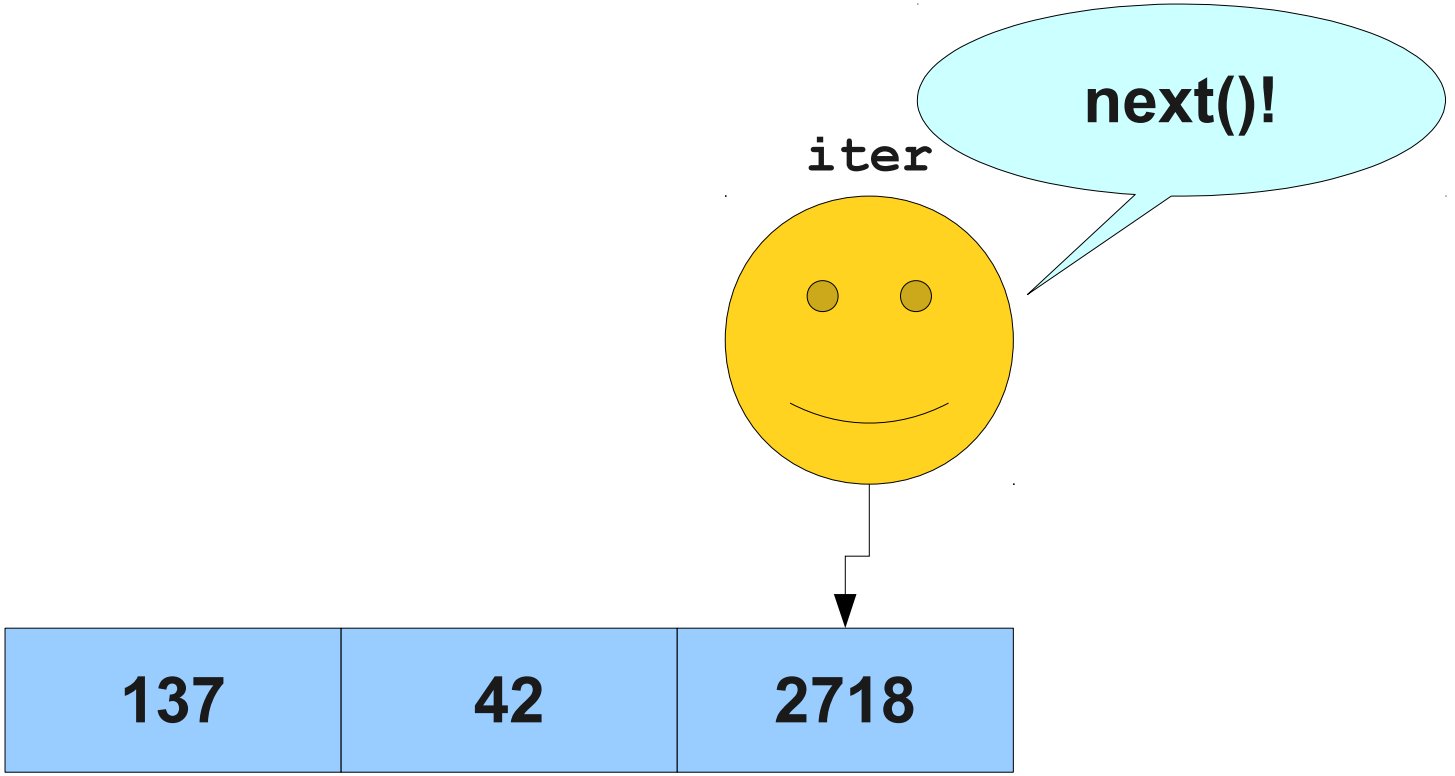



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

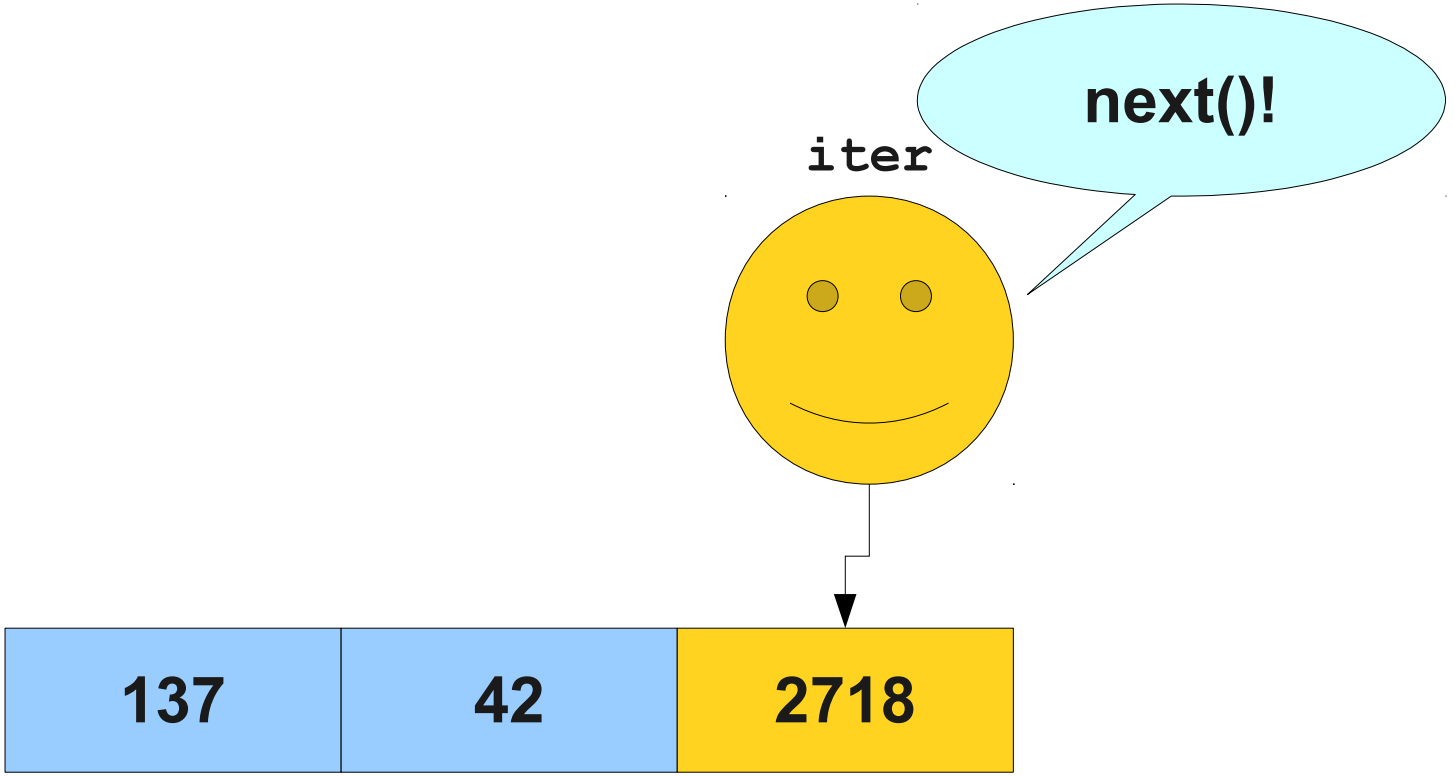



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators

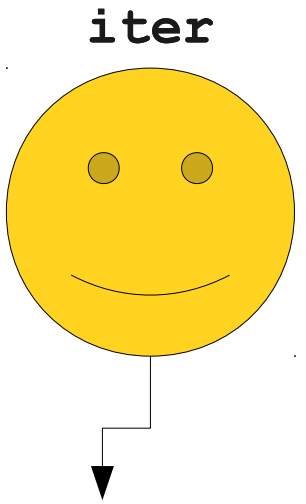



```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators



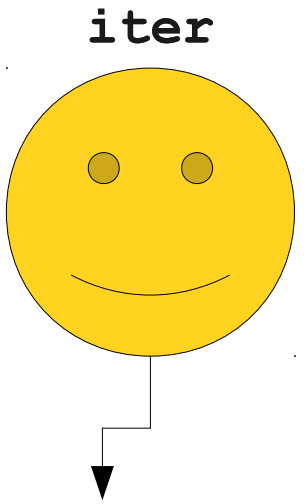
137	42	2718

```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators



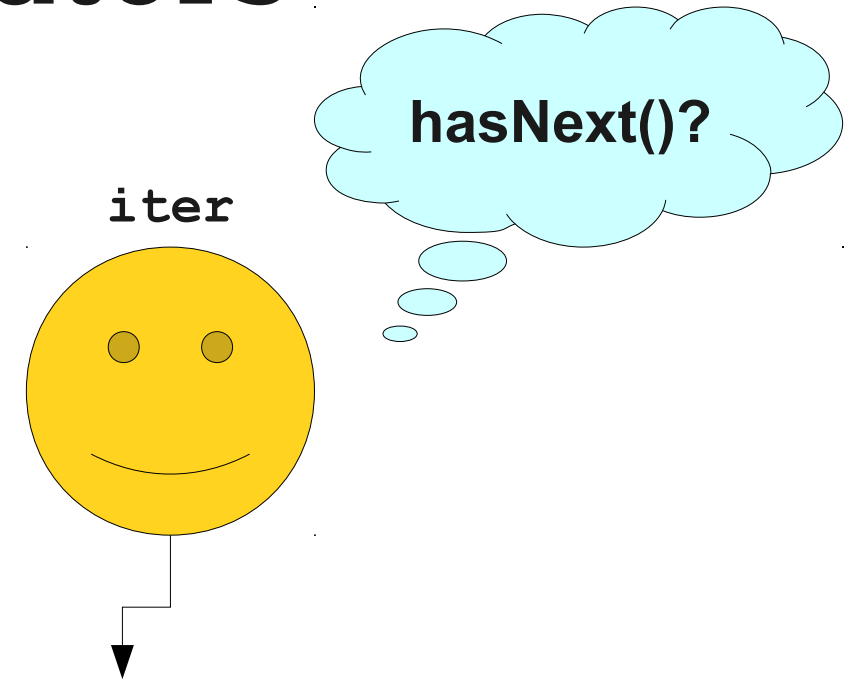
137	42	2718

```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators



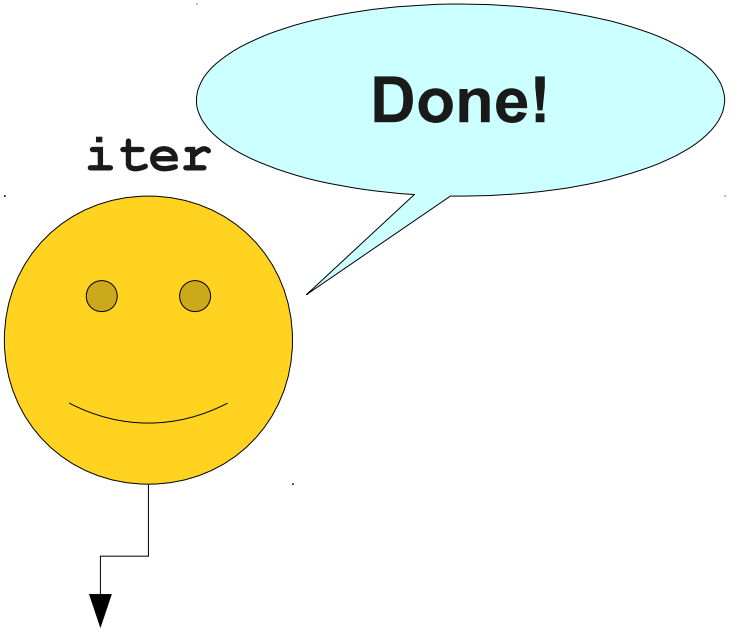
137	42	2718

```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators



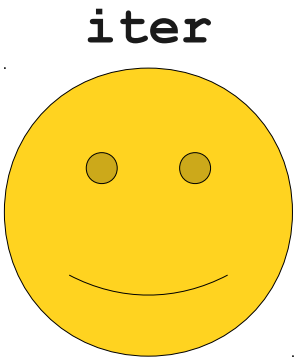
137	42	2718

```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Java Iterators



137	42	2718

```
ArrayList<Integer> myList = /* … */

Iterator<Integer> iter = myList.iterator();
while (iter.hasNext()) {
    int curr = iter.next();

    /* … use curr … */
}
```

Why Iterators?

- Buying a cake versus buying cake mix:
  - If you buy a cake, you need to eat it soon.
  - If you buy cake mix, you can eat it at your convenience.
- For loops versus iterators:
  - If you have a for loop, you always iterate across the range at a particular point in the code.
  - If you get back an iterator, you can iterate over it at your convenience.

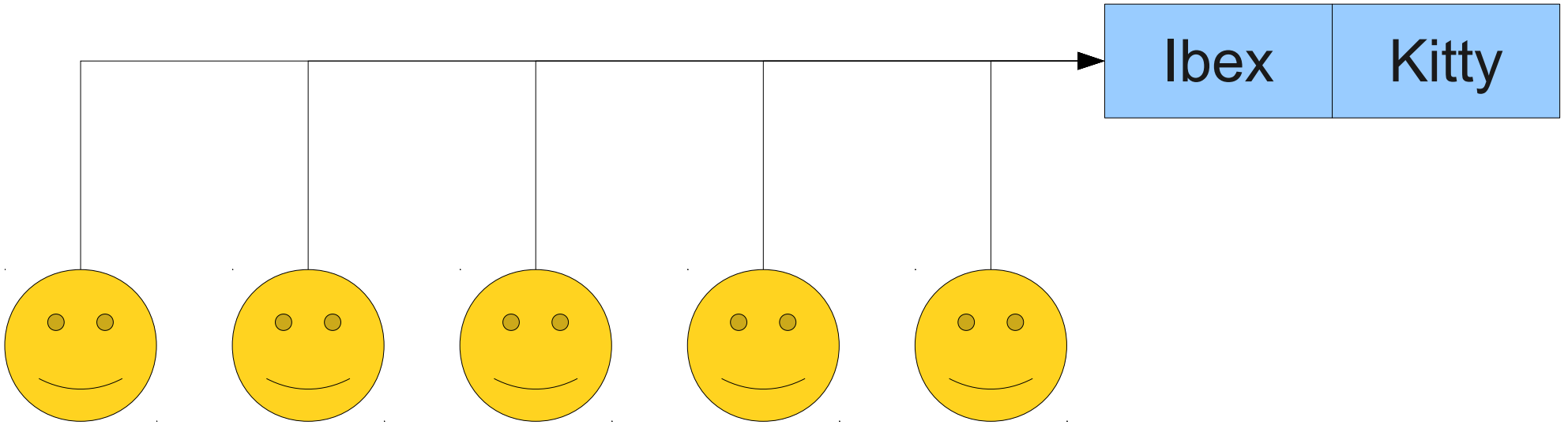
A Word of Warning

A Word of Warning

- The following will loop forever on a nonempty collection:

```
while (collection.iterator().hasNext()) {
    /* … */
}
```

- Every time that you call `.iterator()`, you get back a new iterator to the start of the collection.

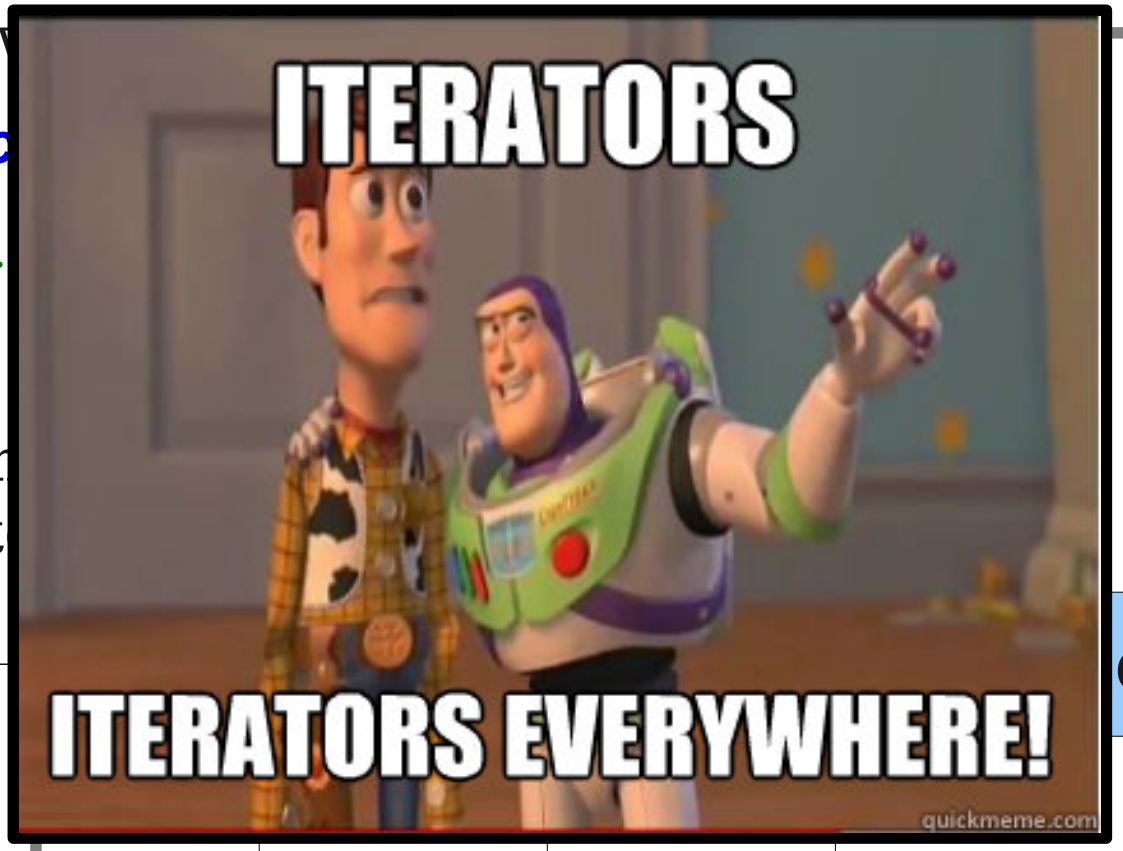


A Word of Warning

- The follo collection:

```
while (c
    /* ..
}
```

- Every tim back a new iterator t



ex | Kitty

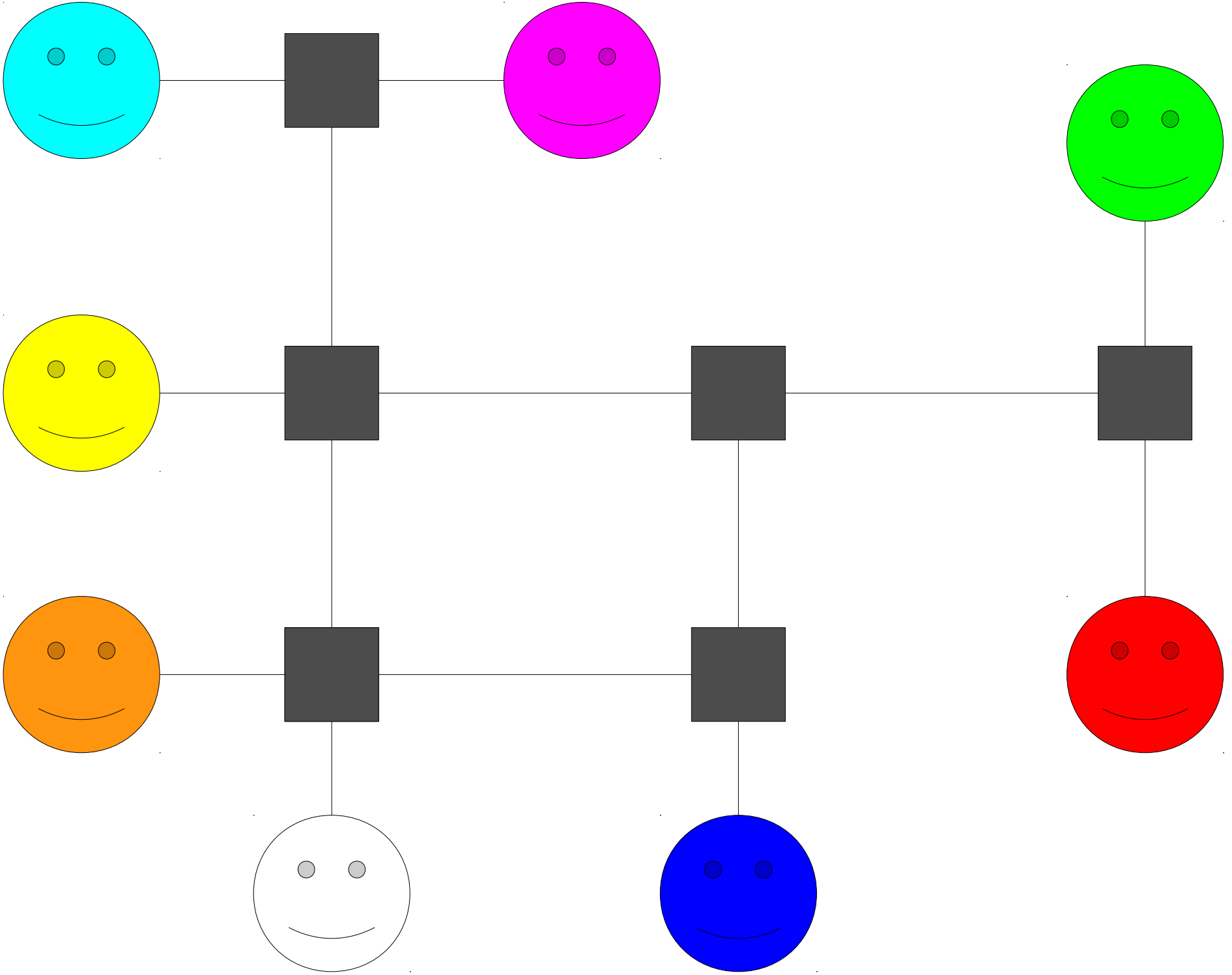
Assignment 7 Demo

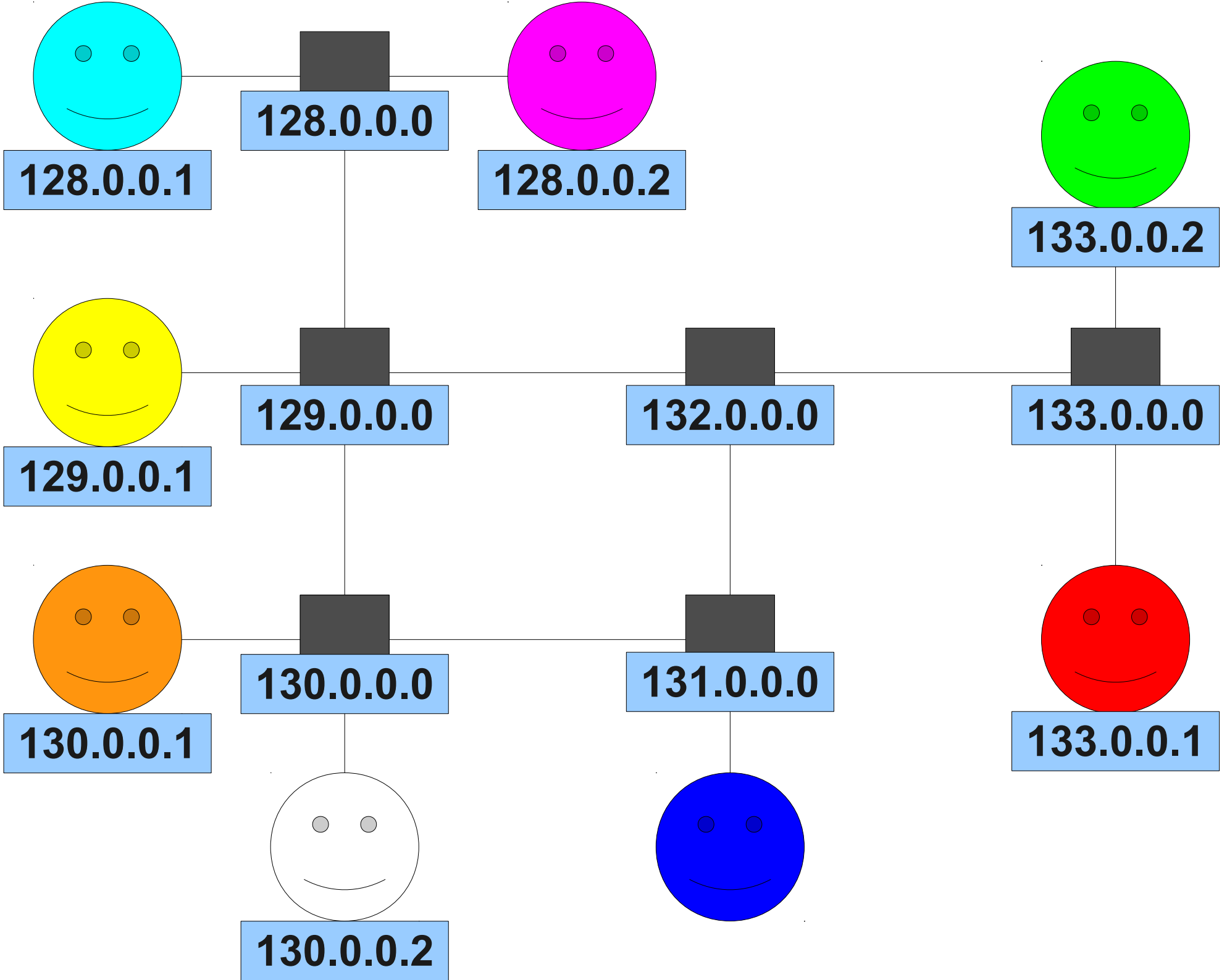
Assignment 7

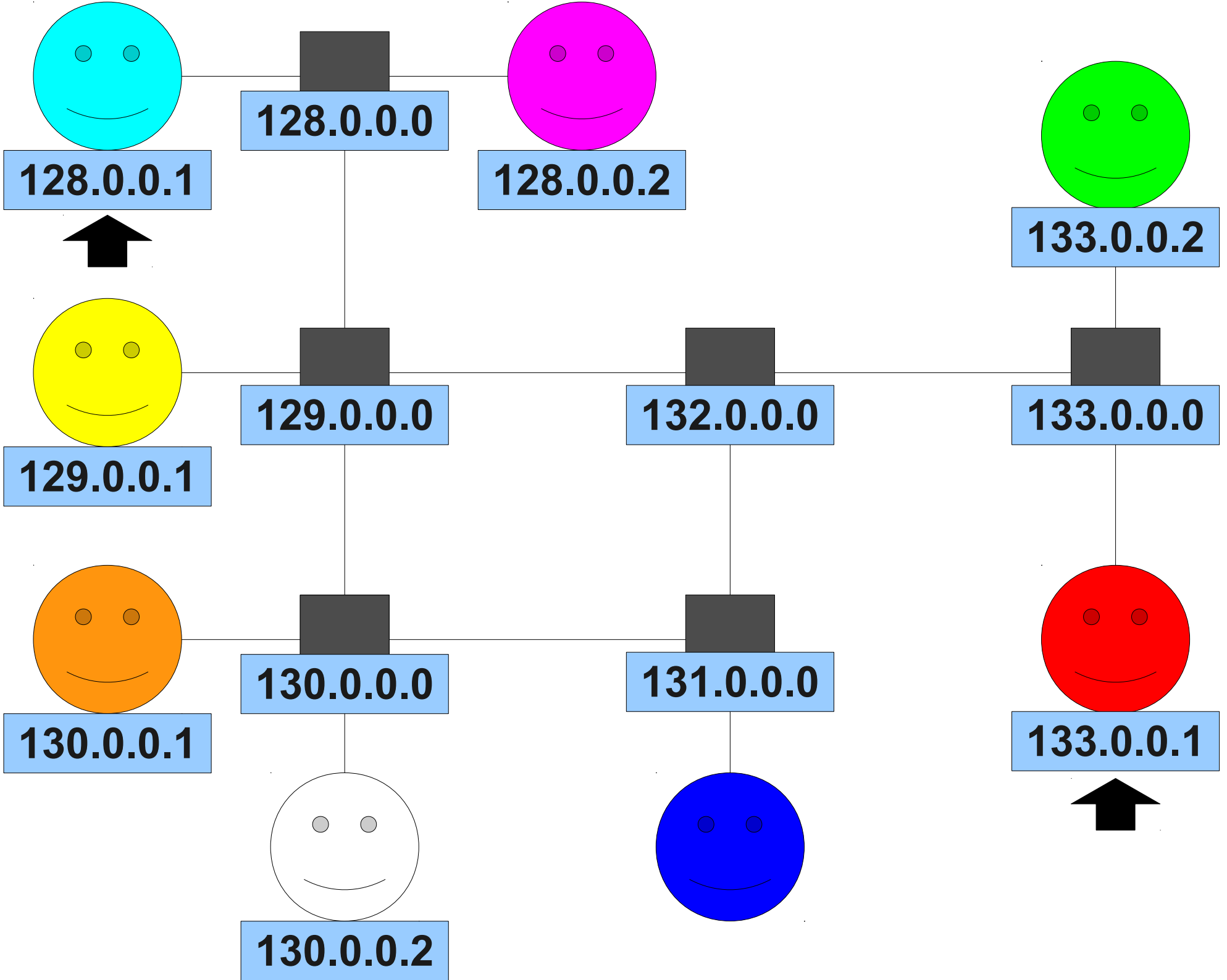
- We're releasing Assignment 7 (FacePamphlet) right now. It's due on Friday, March 21 at 11:30AM.
- Great way to synthesize everything from the course together into one assignment.
- Due date is a hard deadline:
  - **No late days may be used.**
  - **No late submissions accepted.**
- Assignment review hours: Sunday, 7PM – 8PM in Hewlett 200.

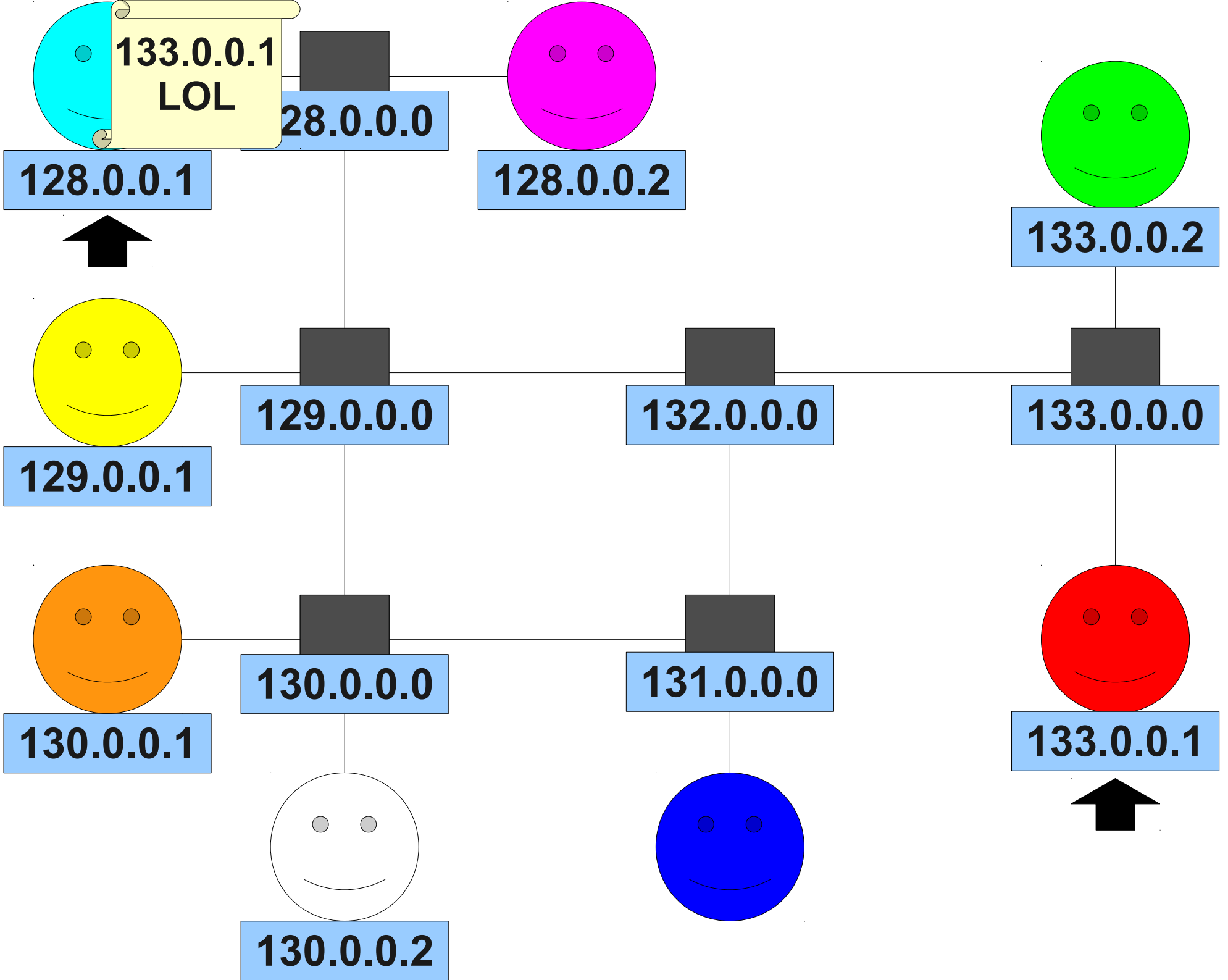
Security and Privacy

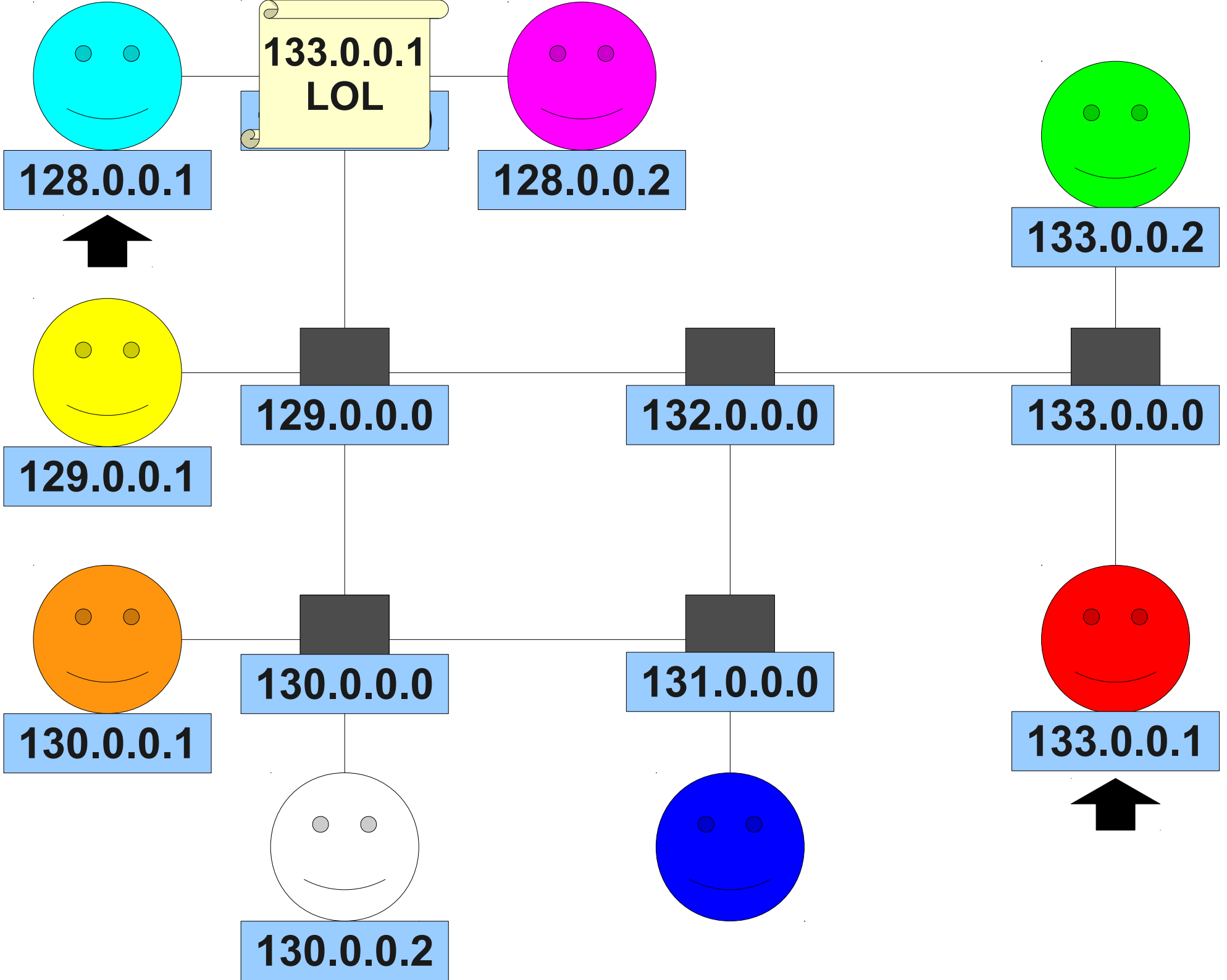
Refresher: How the Internet Works

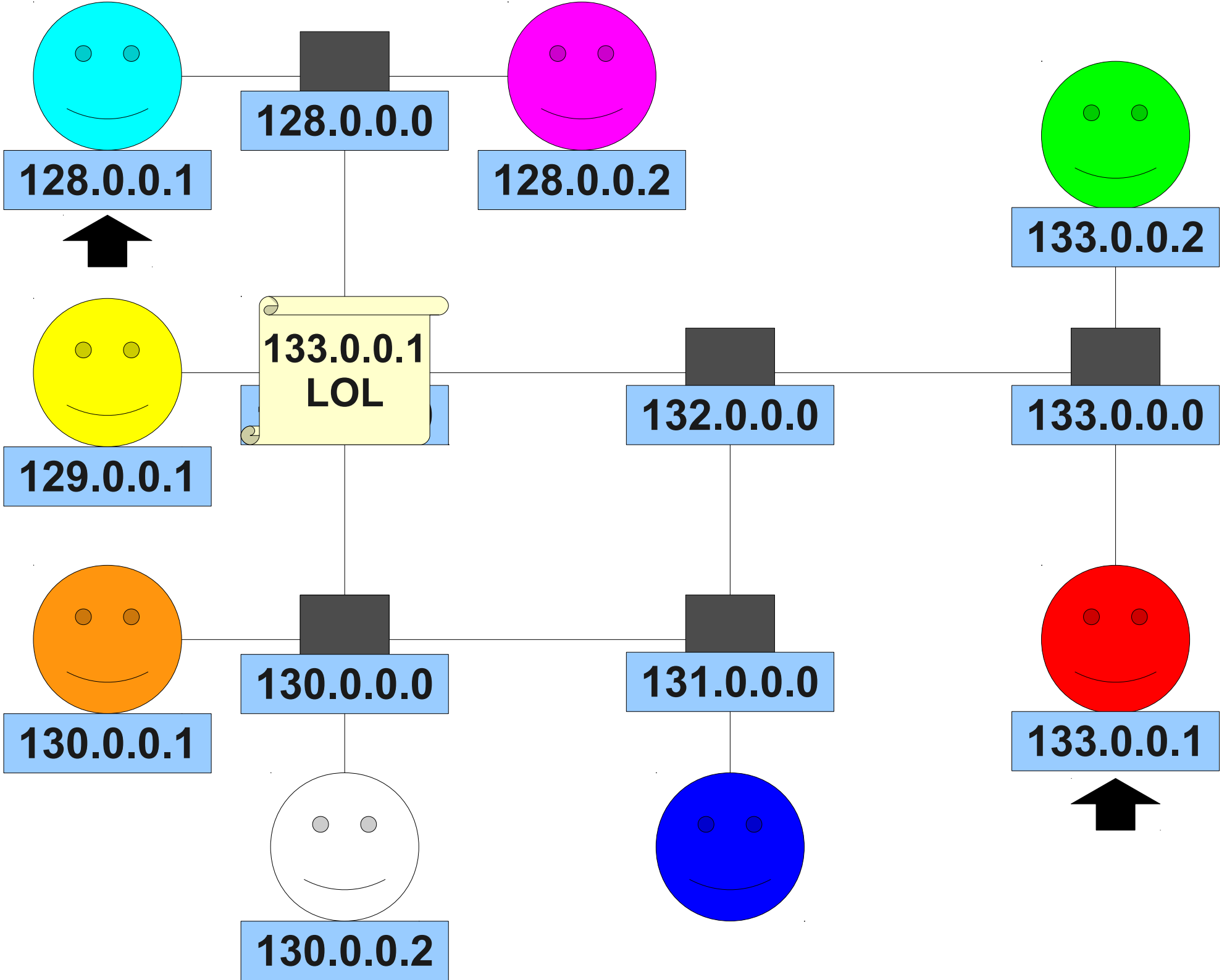


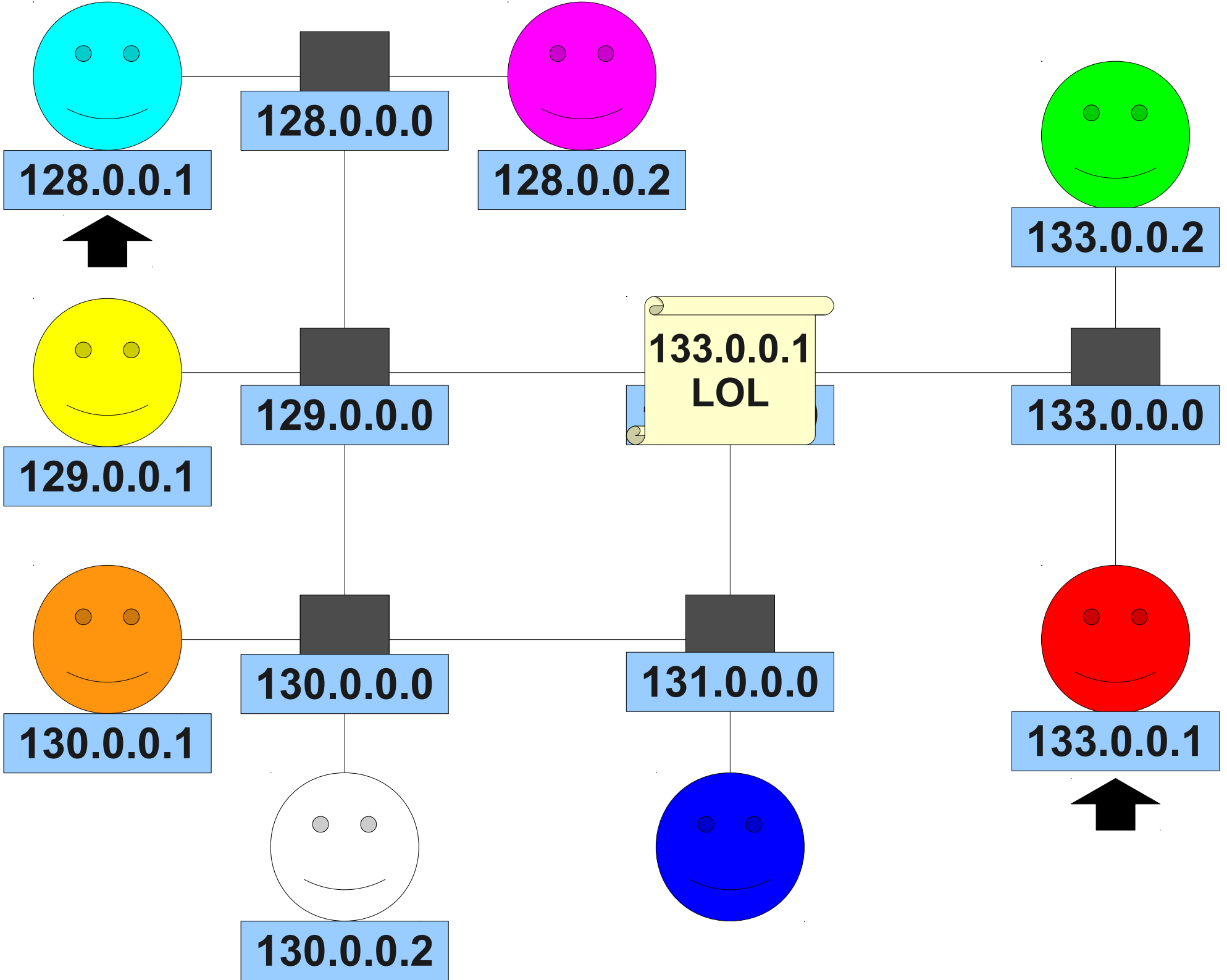

128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
132.0.0.0
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2

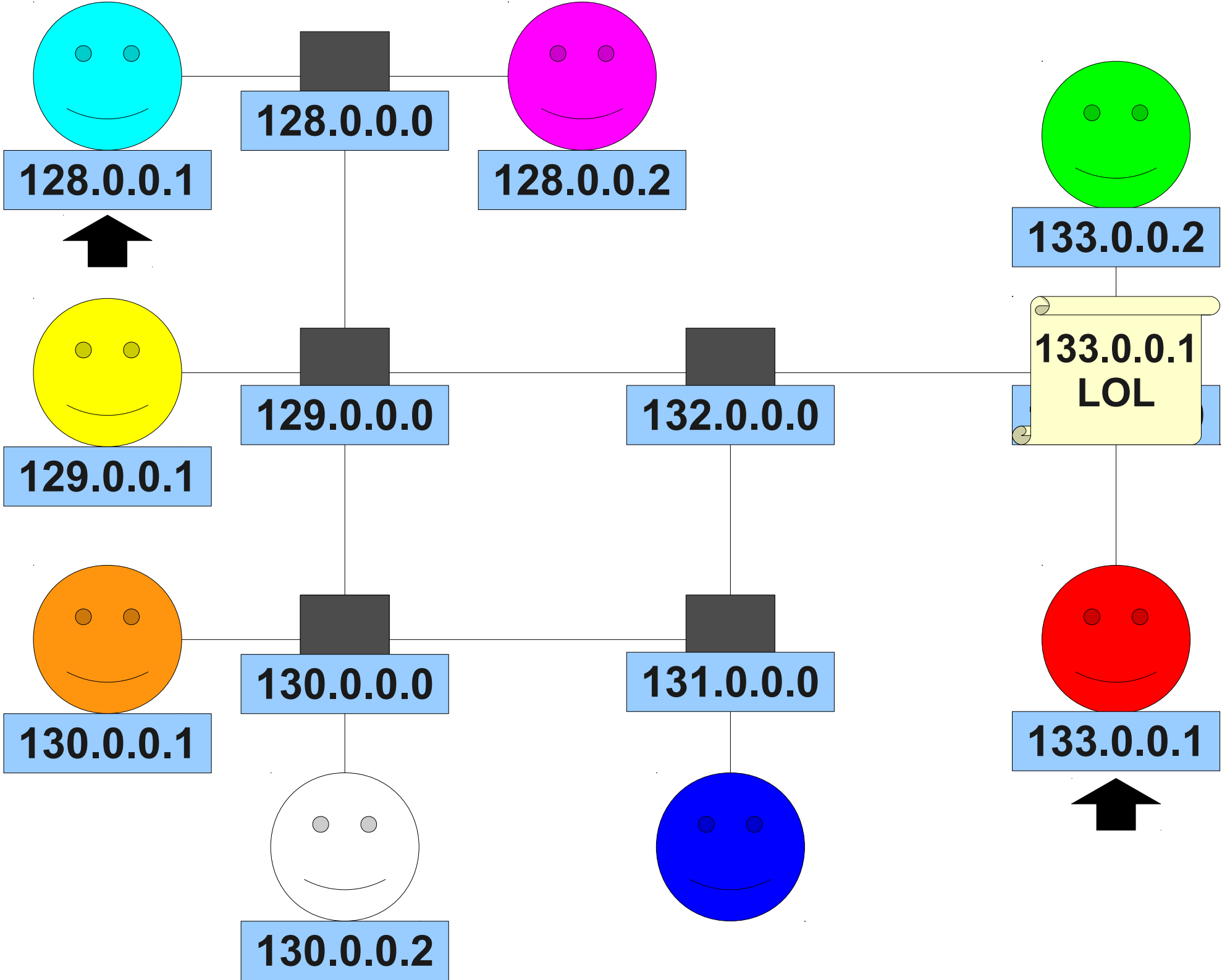

128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
132.0.0.0
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2

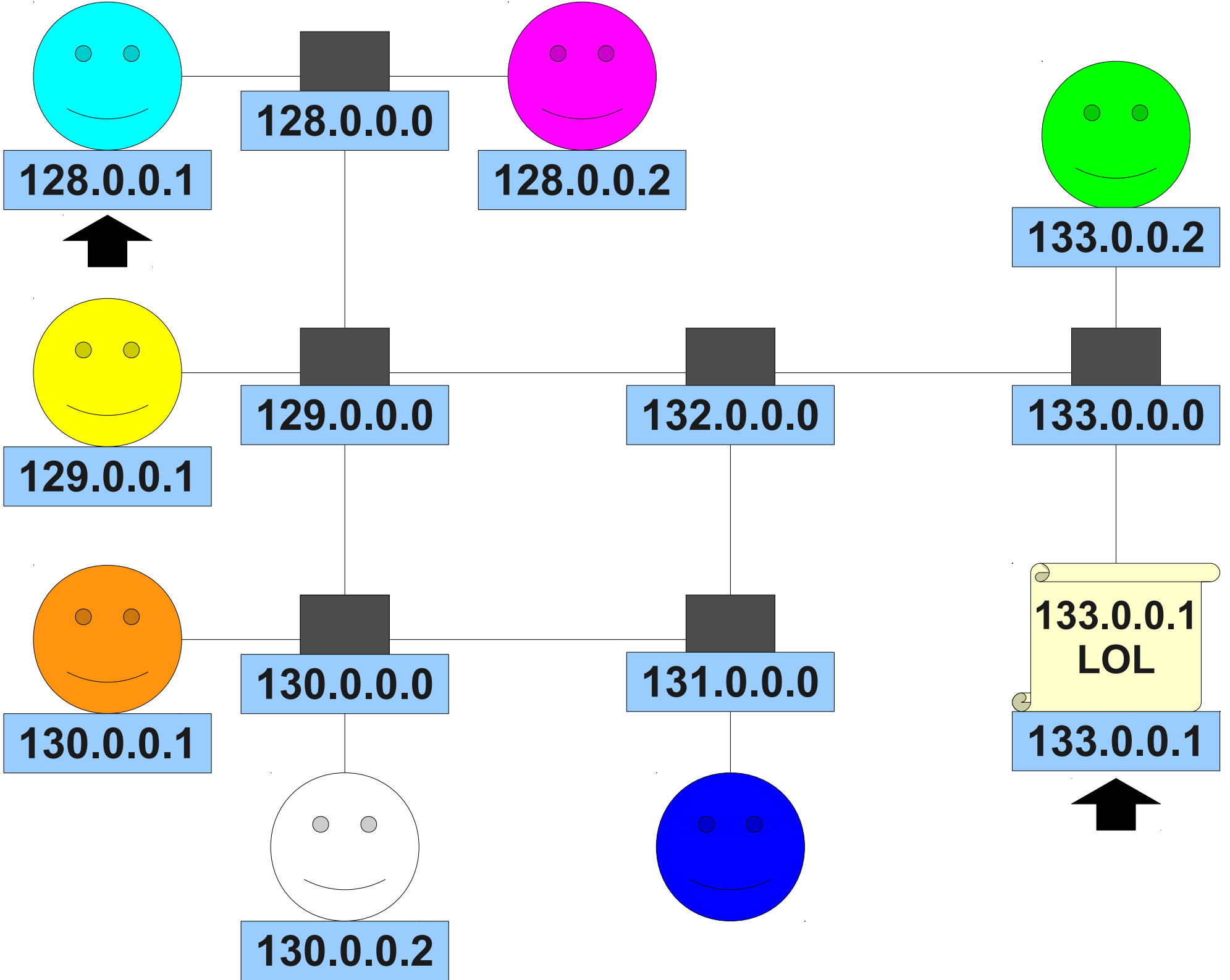

133.0.0.1
LOL
128.0.0.0
128.0.0.1
128.0.0.2
133.0.0.2
129.0.0.0
132.0.0.0
133.0.0.0
129.0.0.1
130.0.0.0
131.0.0.0
130.0.0.1
133.0.0.1
130.0.0.2

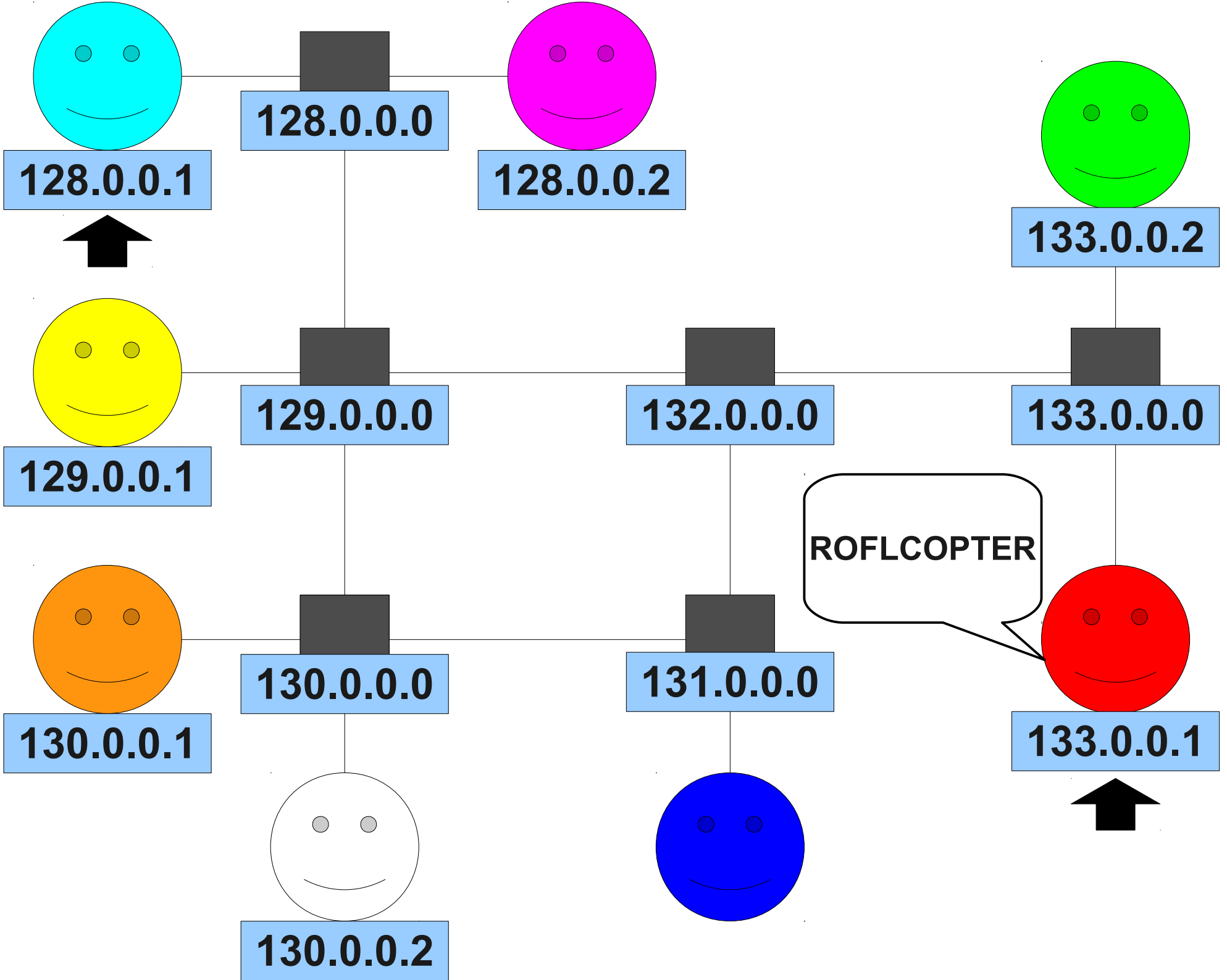

133.0.0.1
LOL
128.0.0.1
128.0.0.2
133.0.0.2
129.0.0.0
132.0.0.0
133.0.0.0
129.0.0.1
130.0.0.0
131.0.0.0
130.0.0.1
133.0.0.1
130.0.0.2

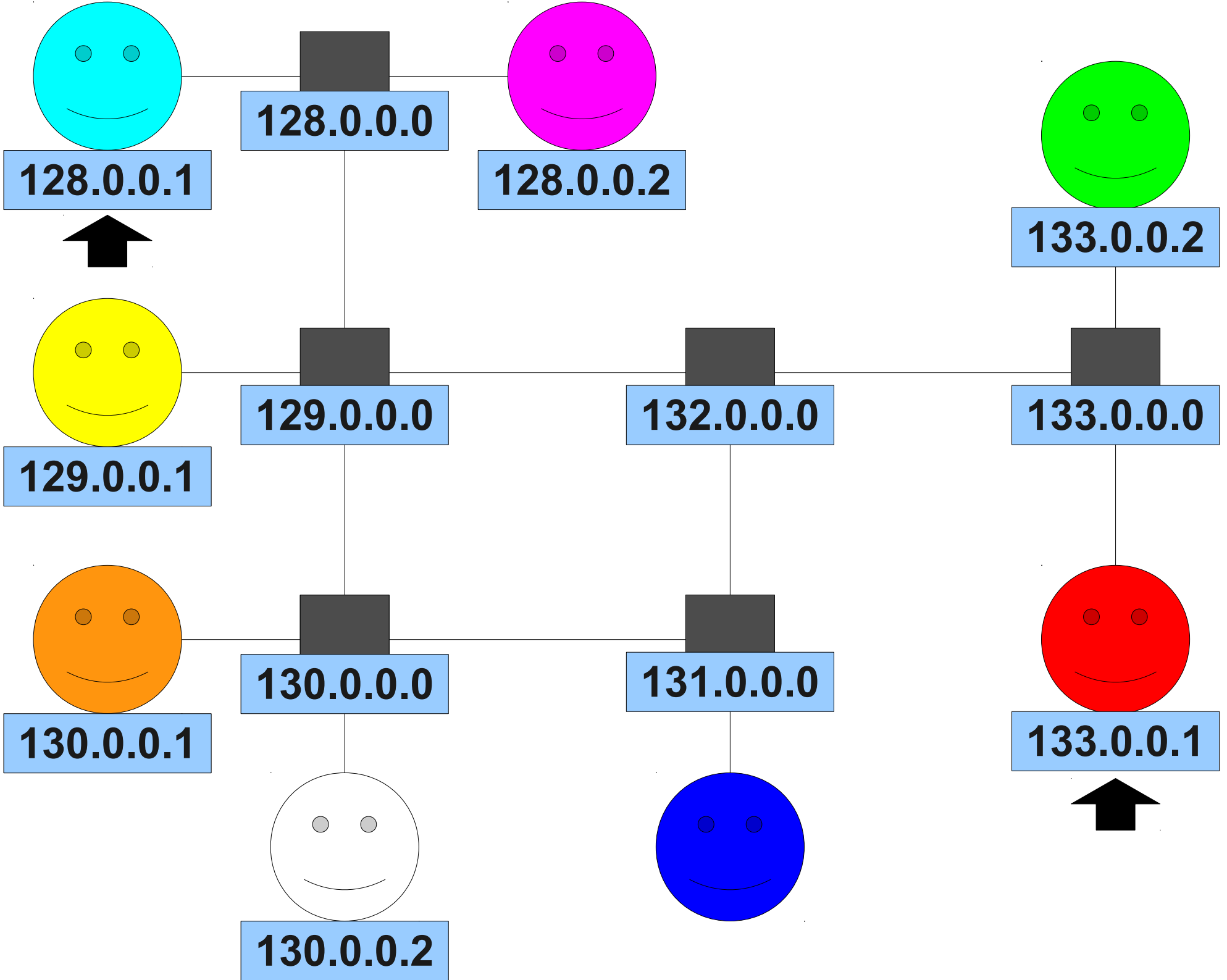

128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
133.0.0.1
LOL
132.0.0.0
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2

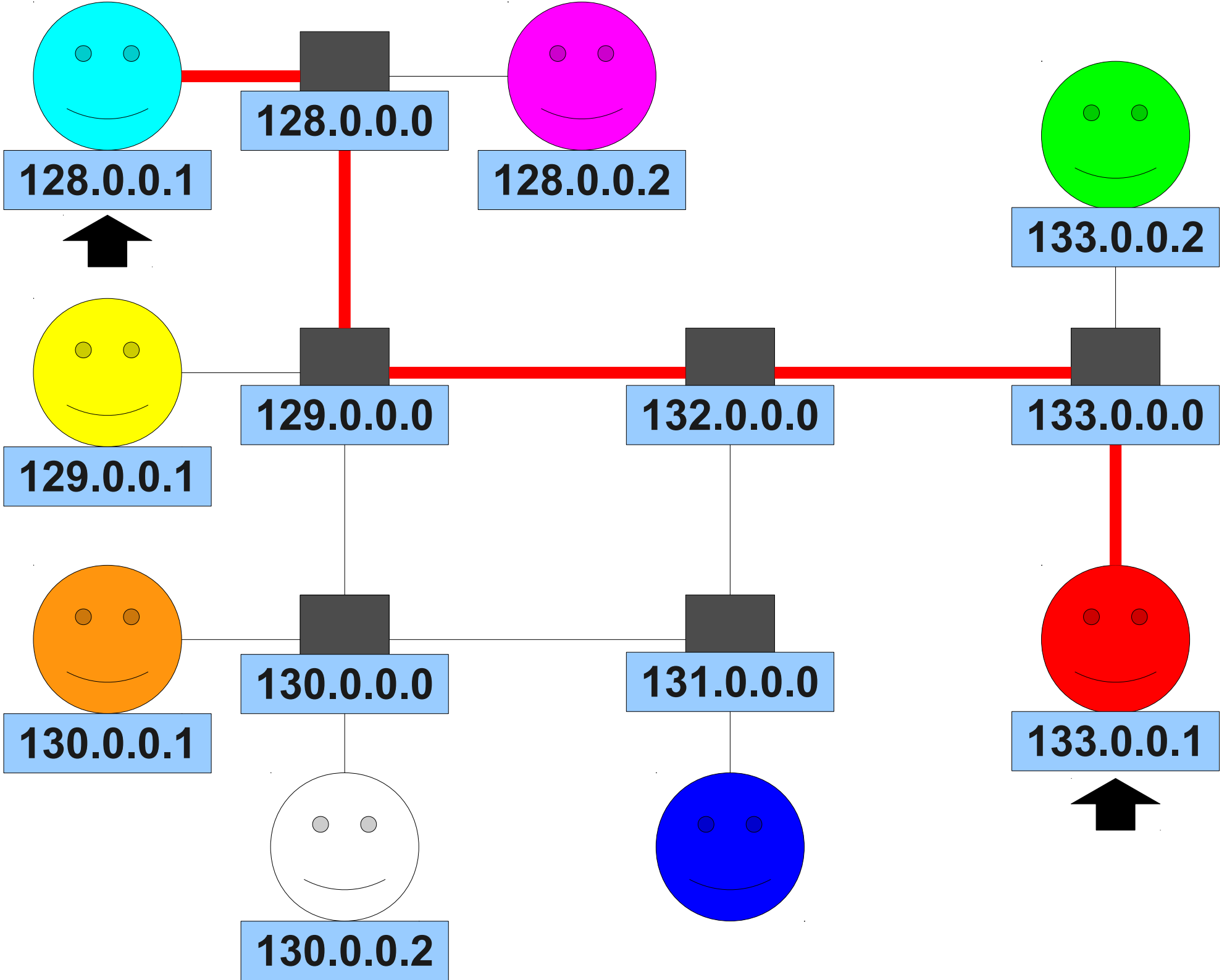

128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
133.0.0.1
LOL
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2


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

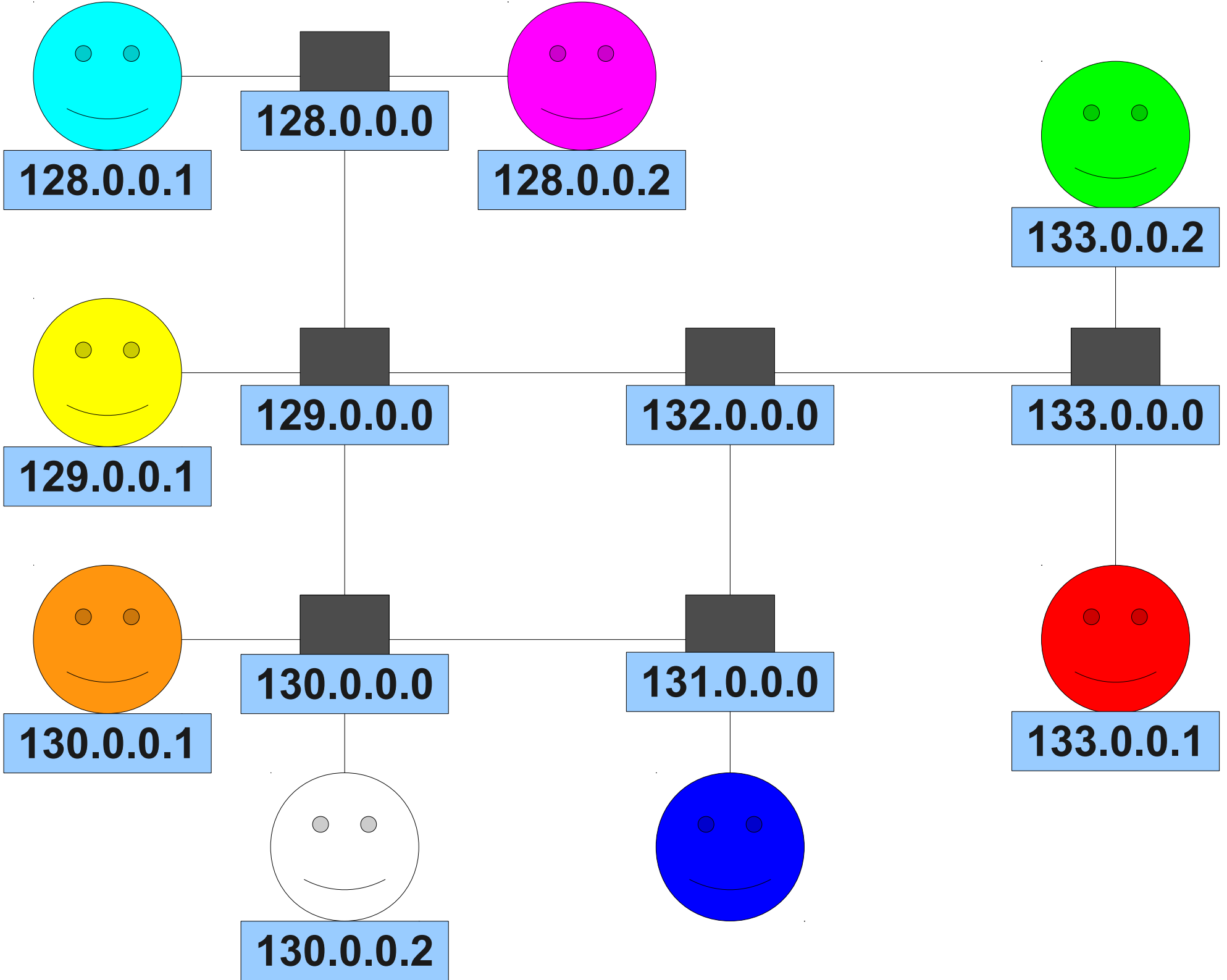

128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
132.0.0.0
133.0.0.0
ROFLCOPTER
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2

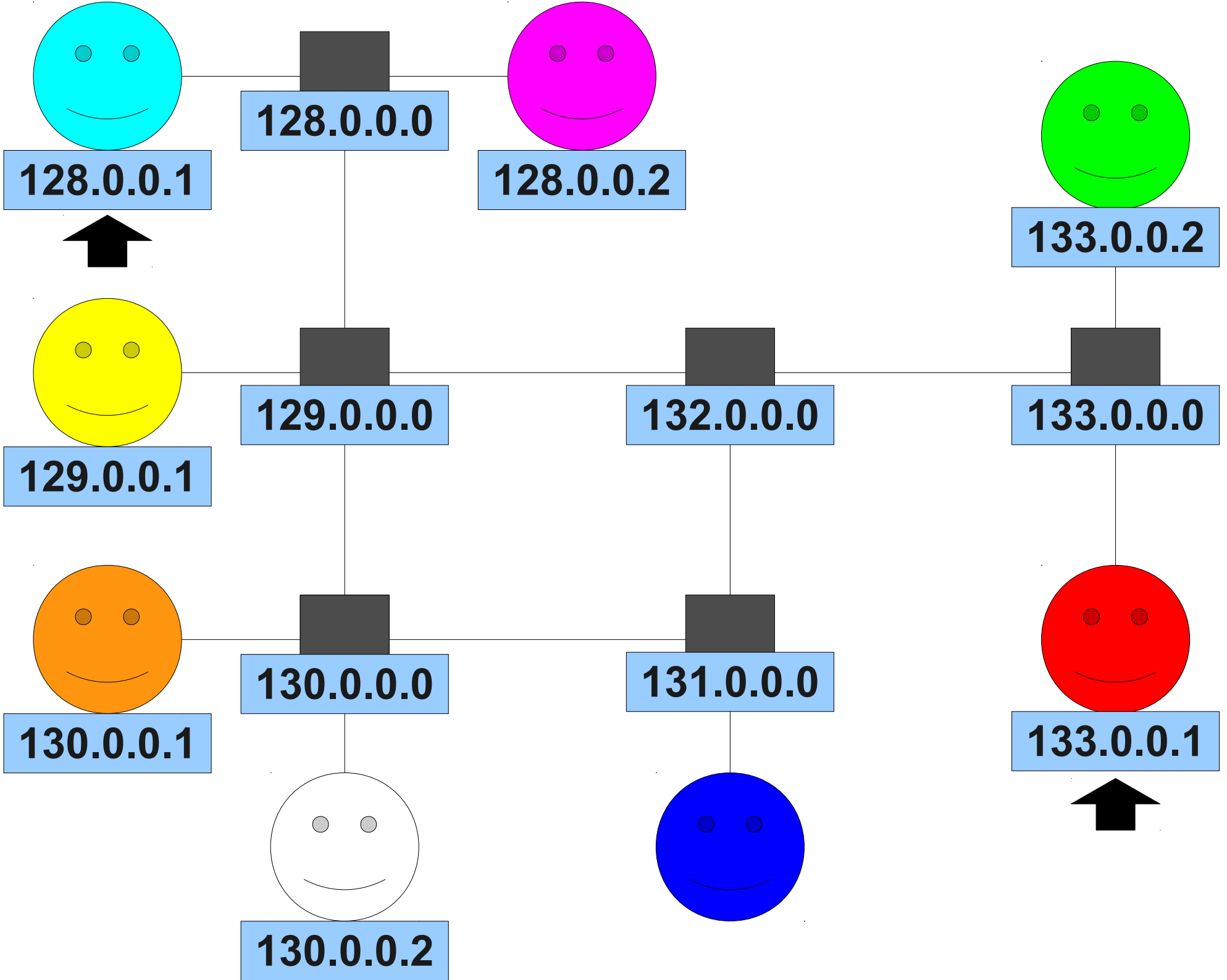

128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
132.0.0.0
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2

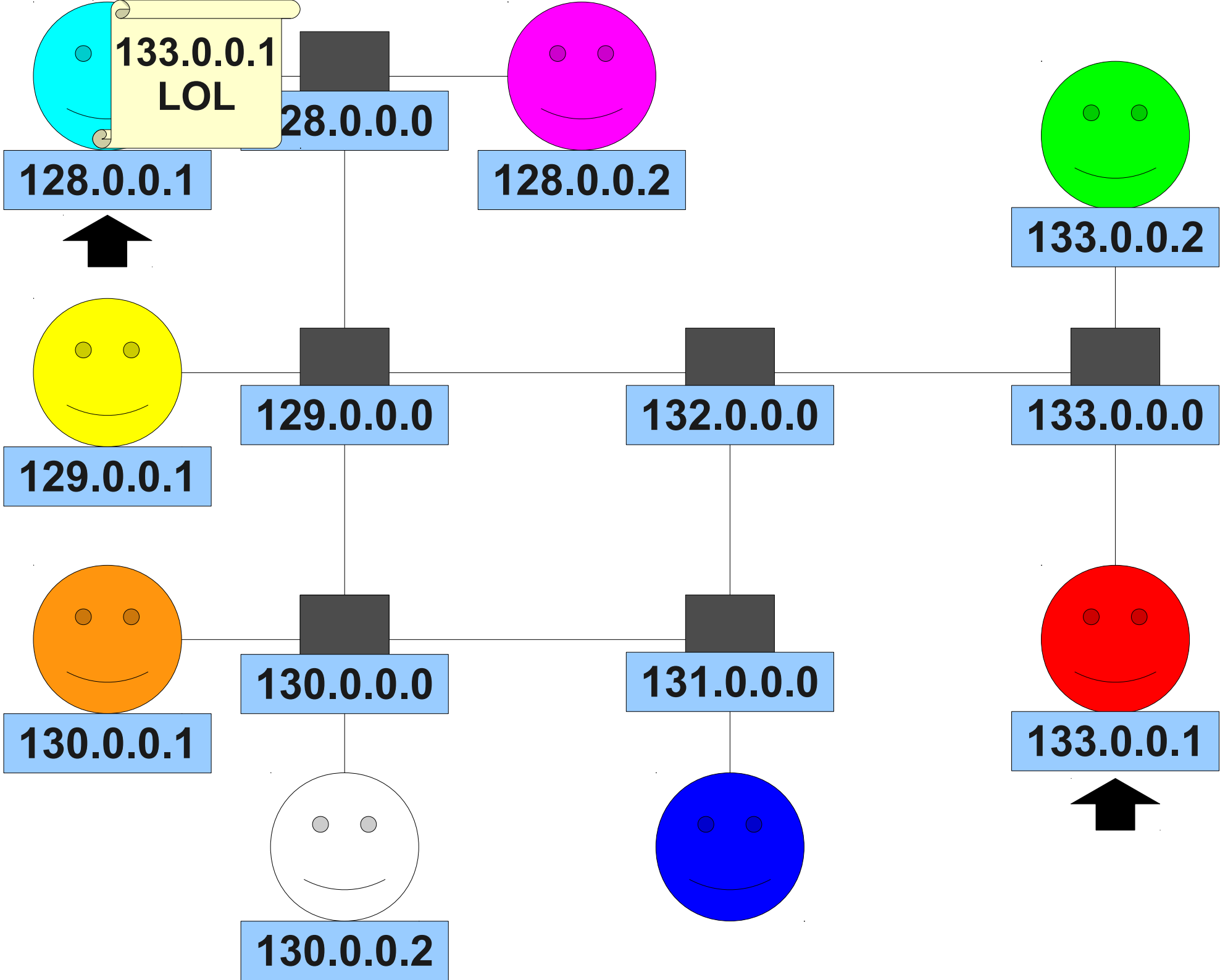

128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
132.0.0.0
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2

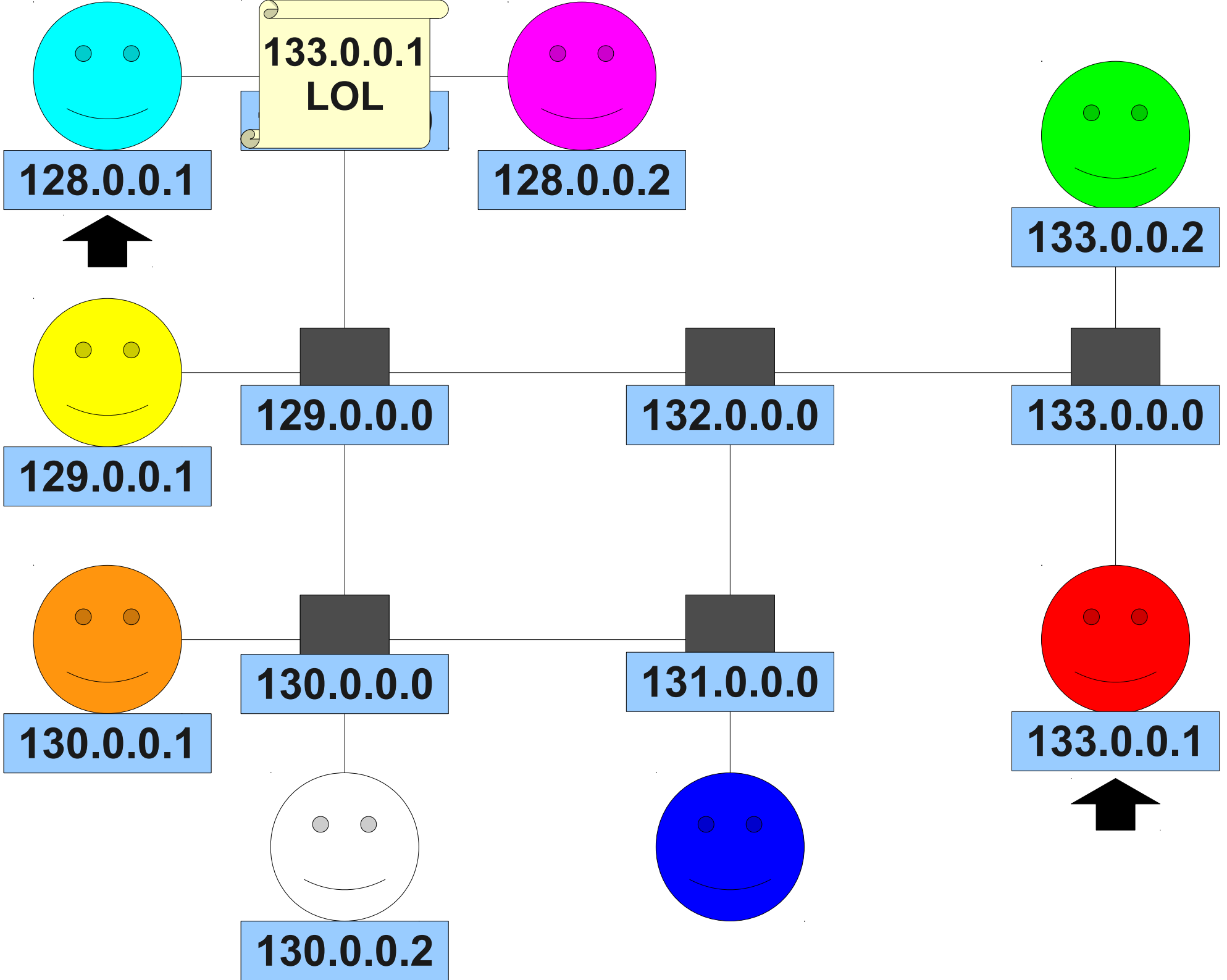
Sending Secrets

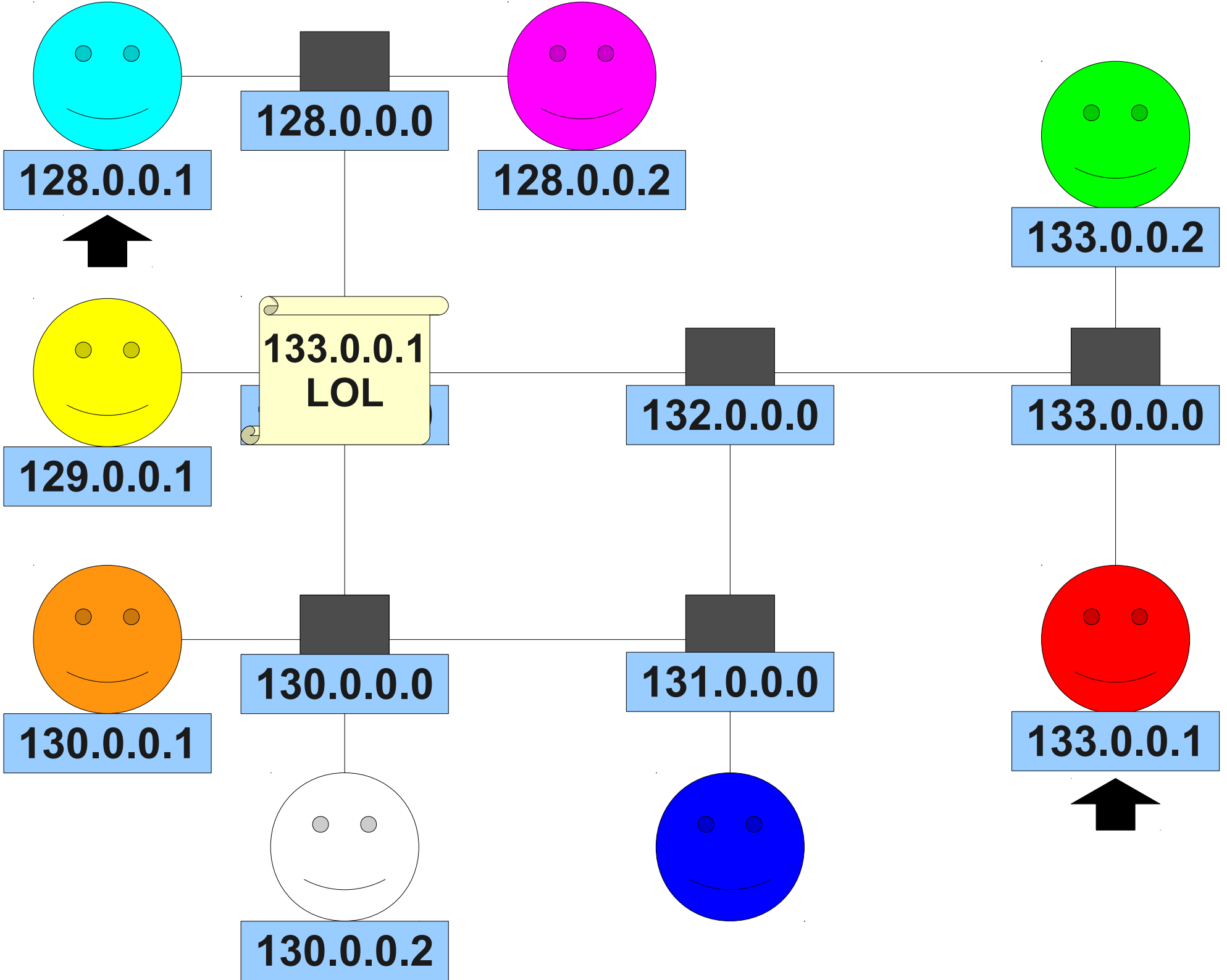
- When you send a message from one computer to another over the Internet, any of the routers between you and your destination can read that message.
- To send secrets, the data is typically **encrypted**; only the receiver can read it and it looks like gibberish to everyone else.
- **Advice:** Use `https` instead of `http` any time you enter a password online.

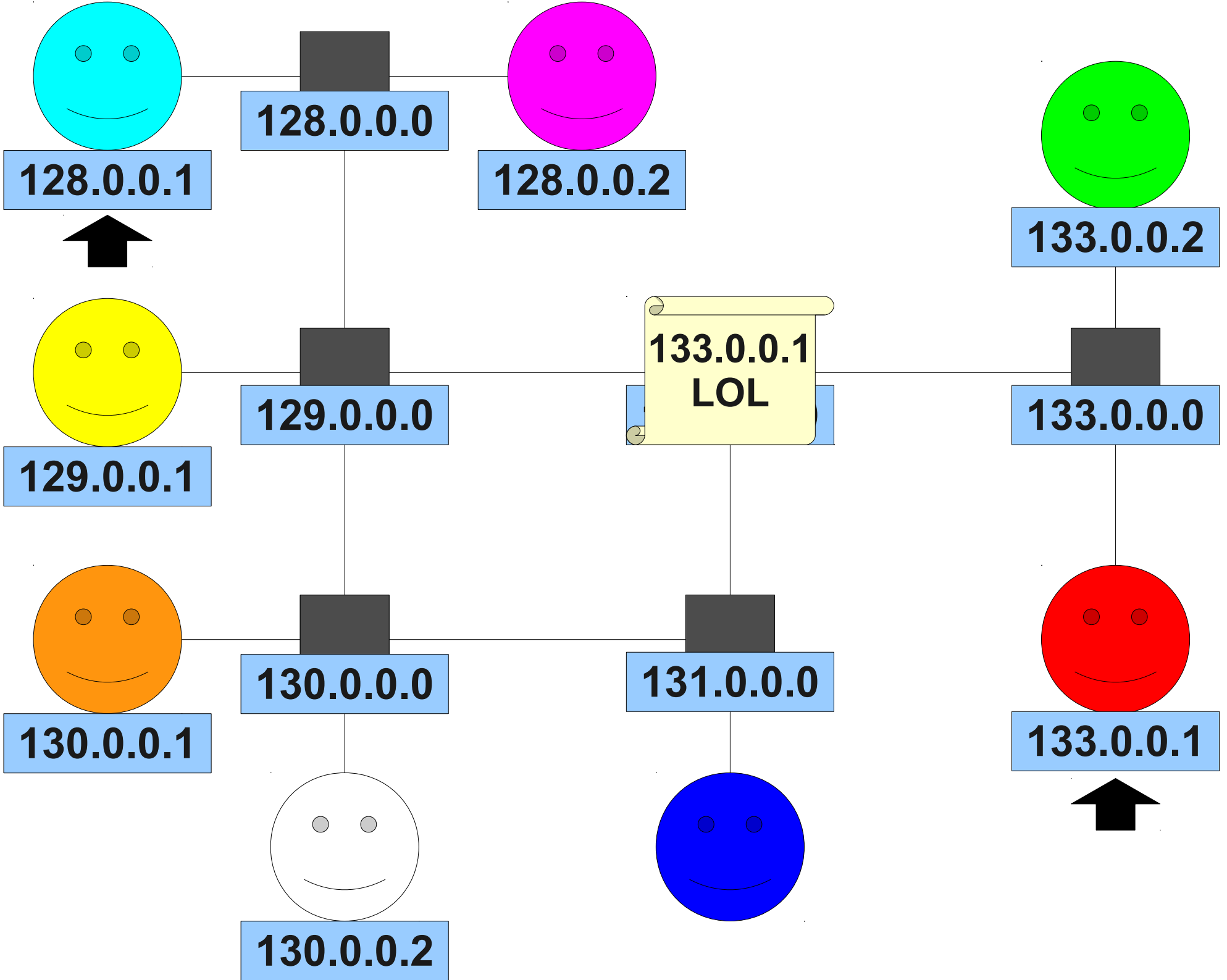

128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
132.0.0.0
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2

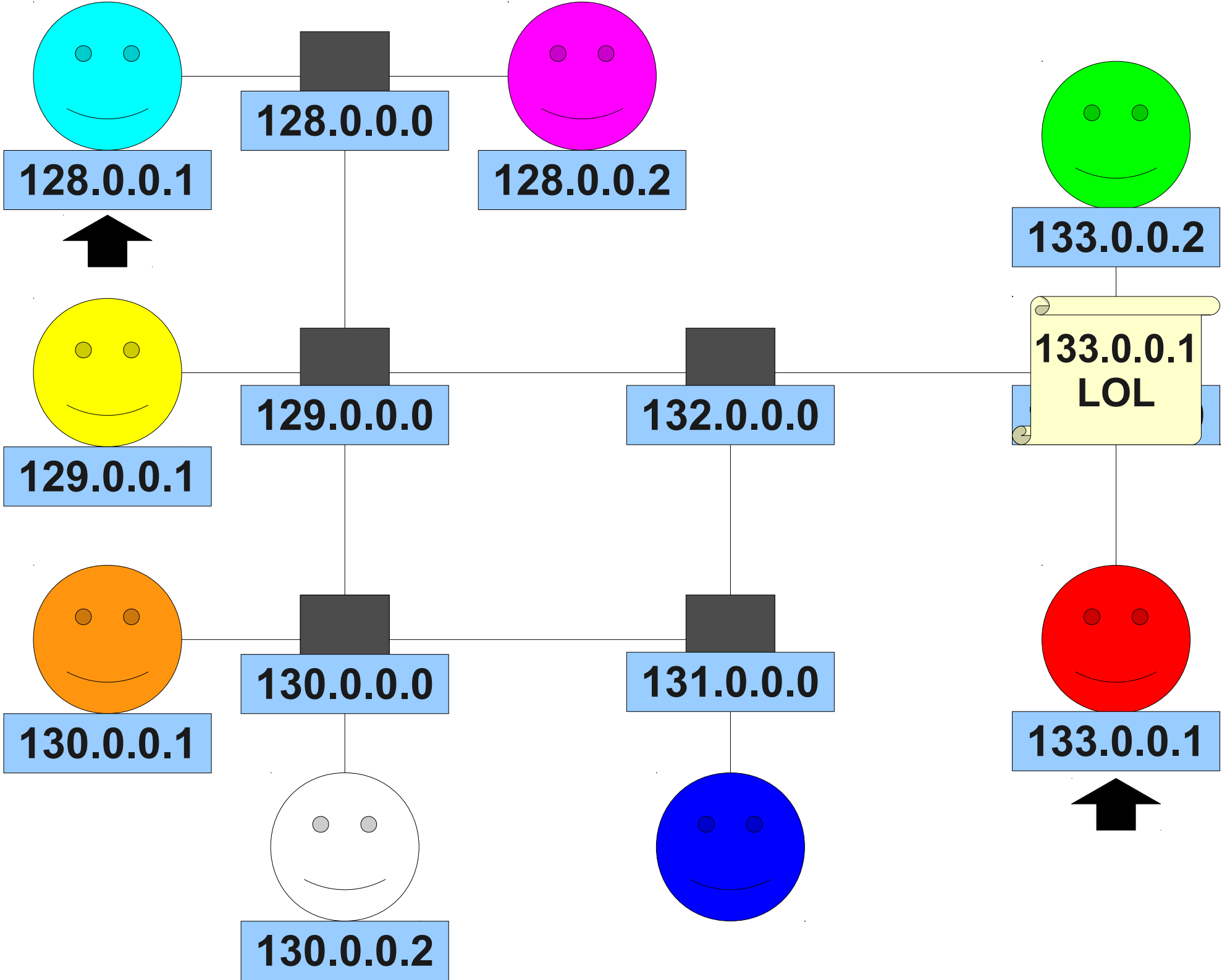

128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
132.0.0.0
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2

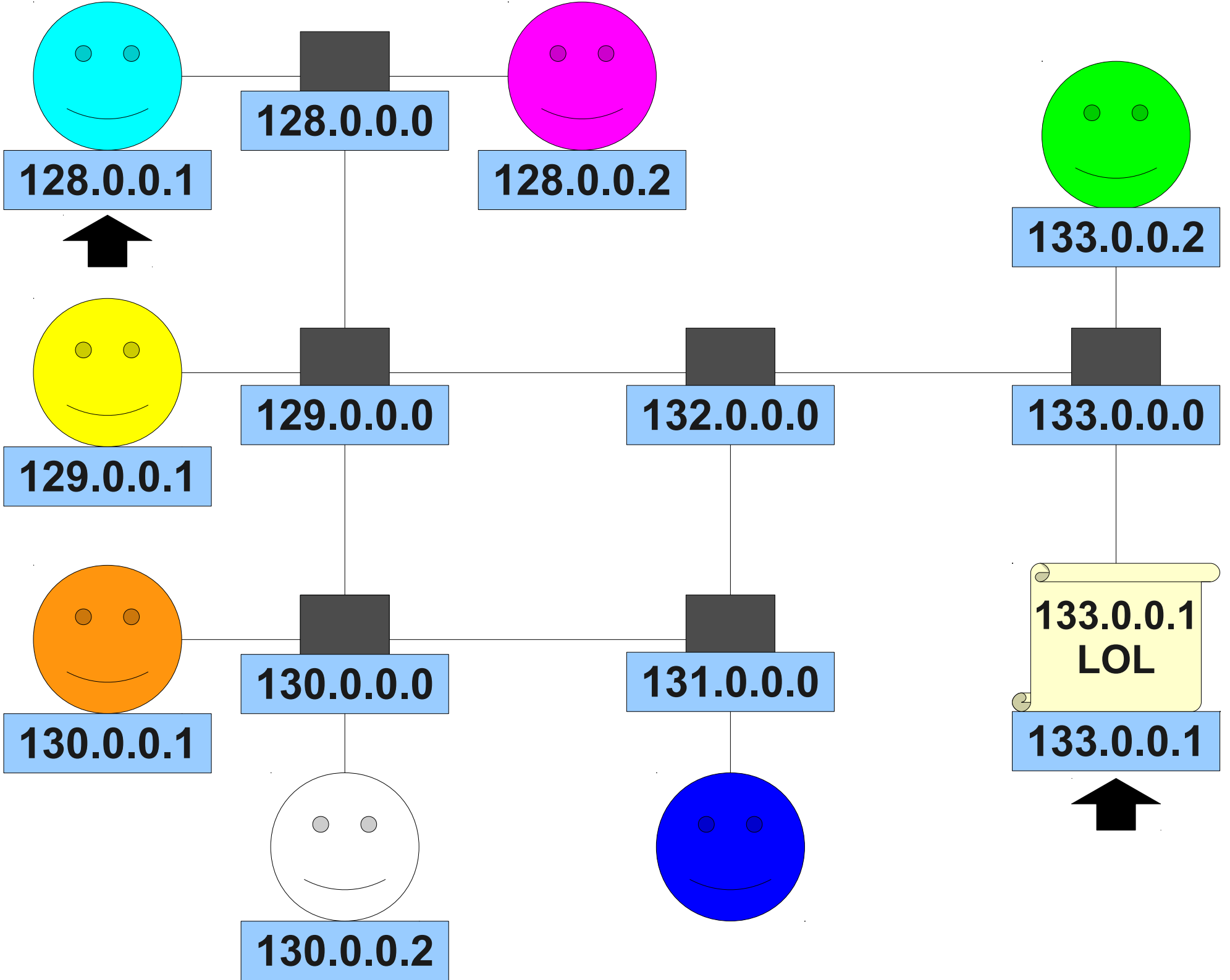

133.0.0.1
LOL
128.0.0.0
128.0.0.1
128.0.0.2
133.0.0.2
129.0.0.0
132.0.0.0
133.0.0.0
129.0.0.1
130.0.0.0
131.0.0.0
130.0.0.1
133.0.0.1
130.0.0.2

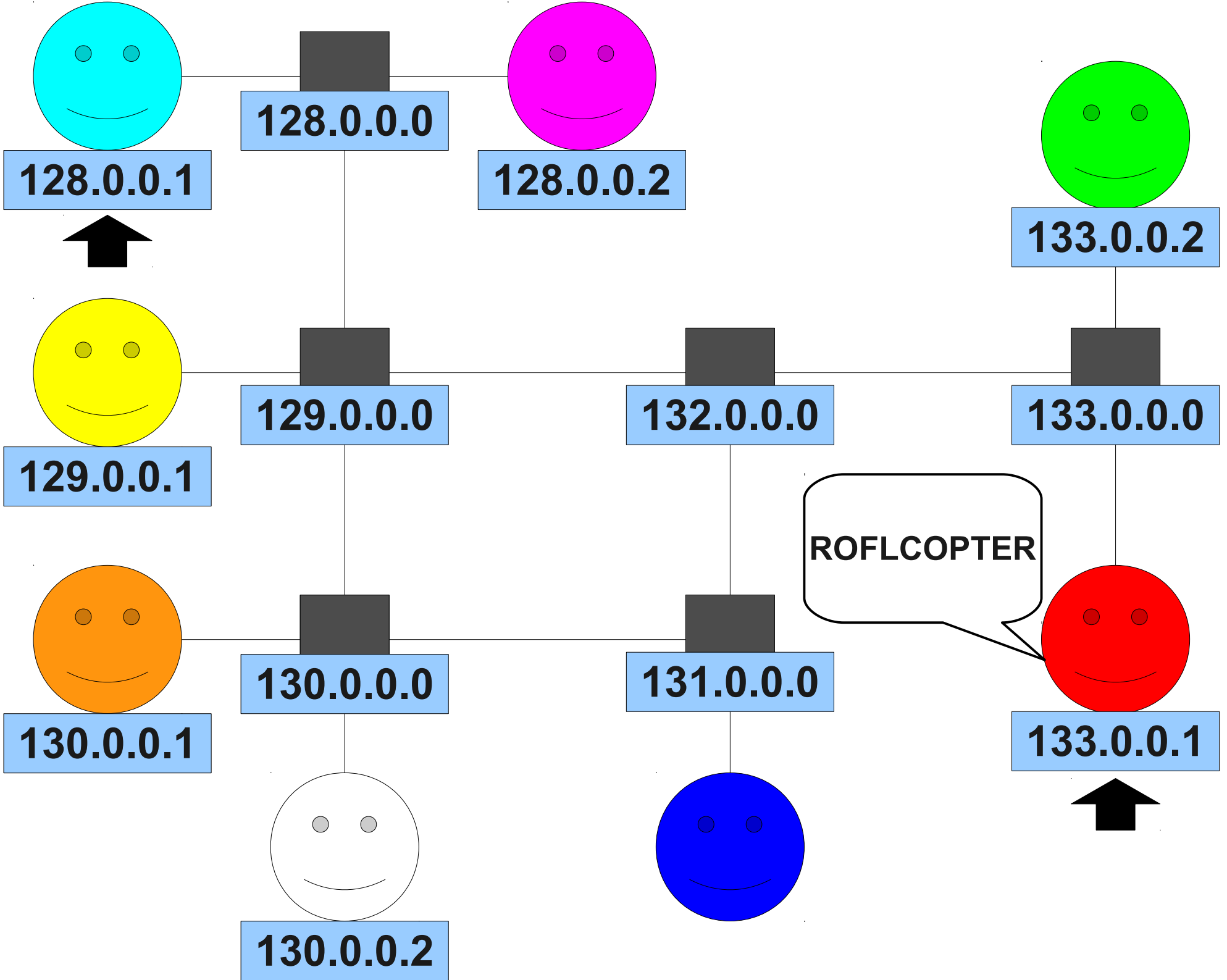

133.0.0.1
LOL
128.0.0.1
128.0.0.2
133.0.0.2
129.0.0.0
132.0.0.0
133.0.0.0
129.0.0.1
130.0.0.0
131.0.0.0
130.0.0.1
133.0.0.1
130.0.0.2

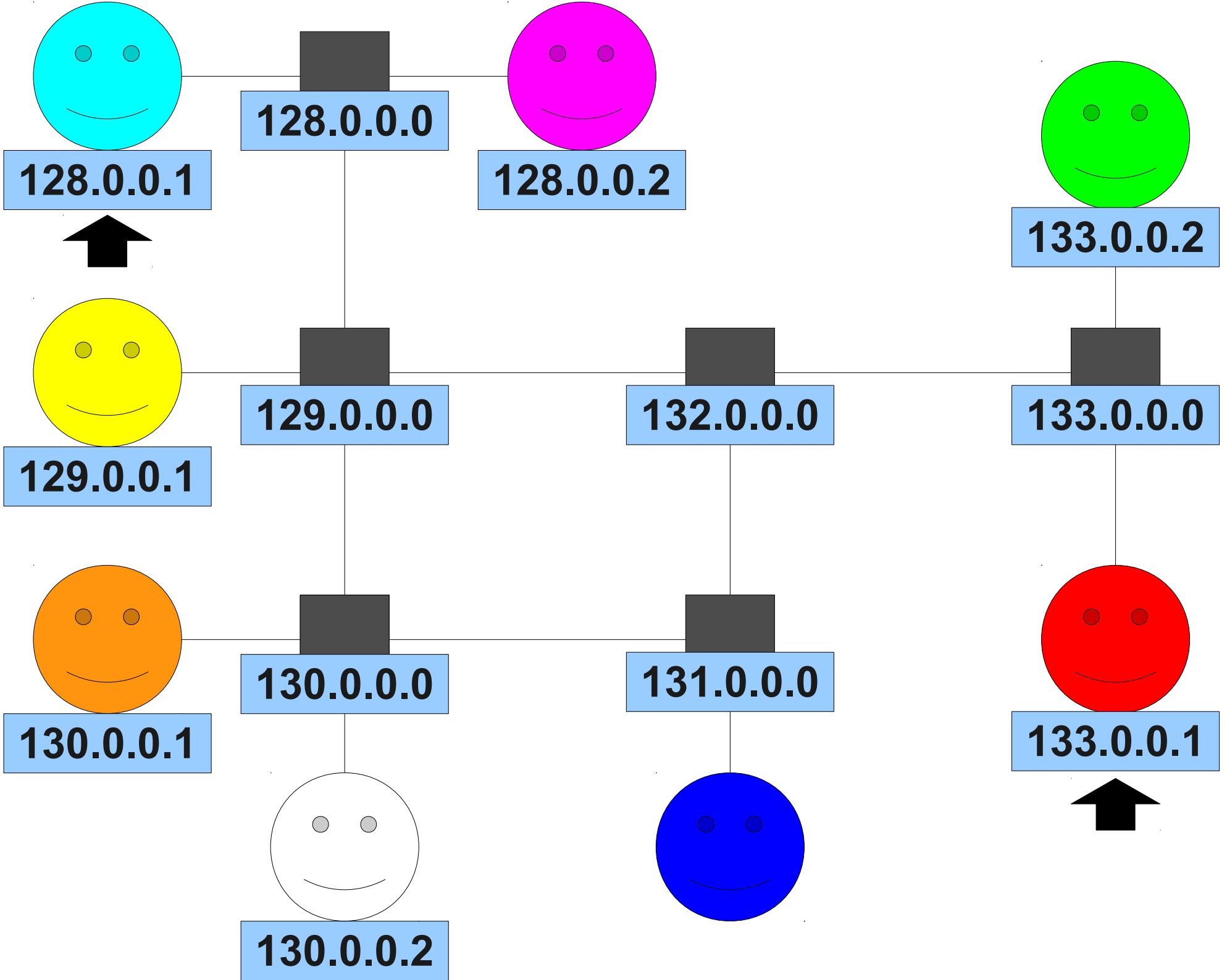

128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
133.0.0.1
LOL
132.0.0.0
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2

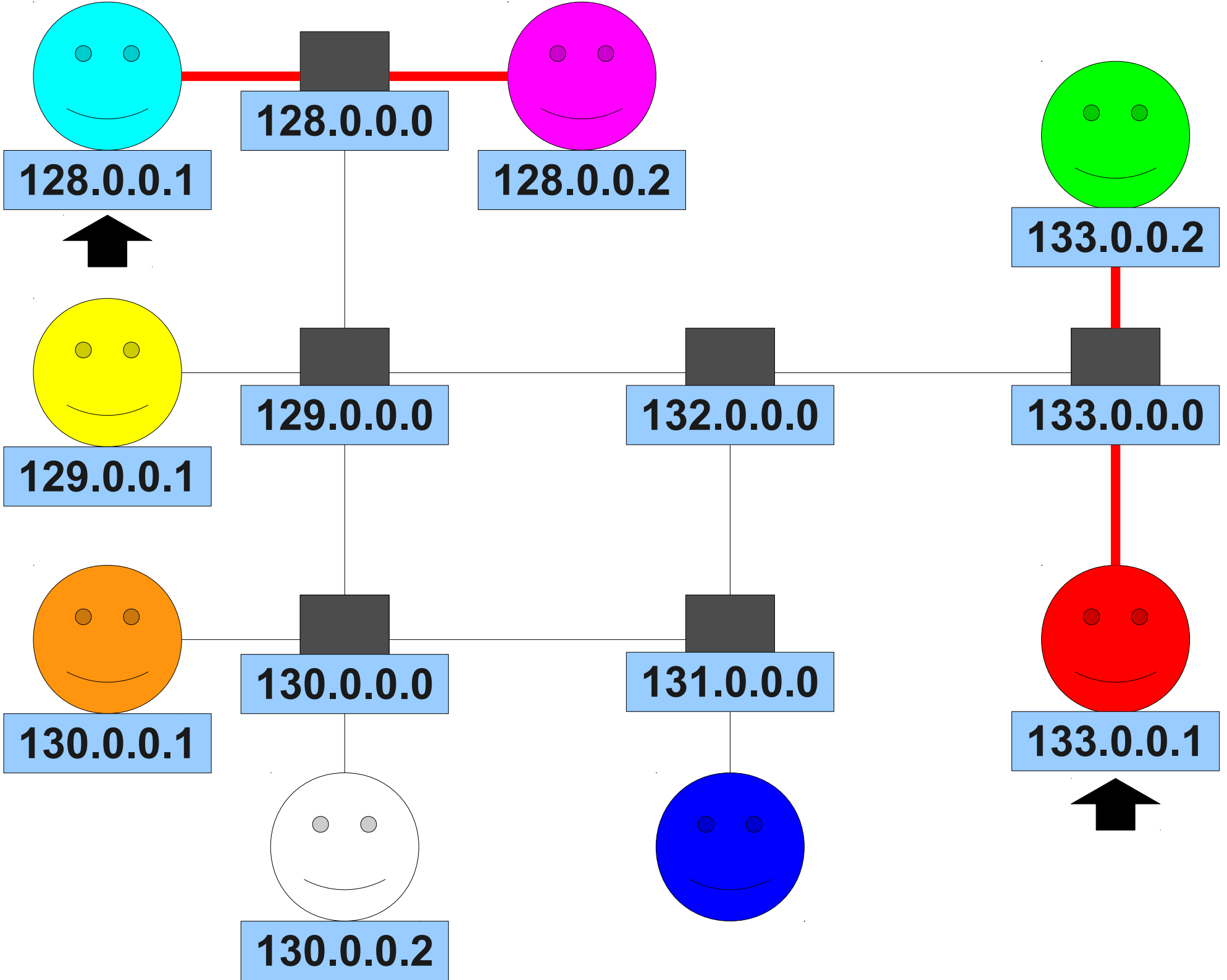

128.0.0.1
128.0.0.0
128.0.0.2
133.0.0.2
129.0.0.1
129.0.0.0
133.0.0.1
LOL
133.0.0.0
130.0.0.1
130.0.0.0
131.0.0.0
133.0.0.1
130.0.0.2


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


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

Ethernet and WiFi

- If you send or receive data over WiFi or Ethernet, that message can be read by anyone else on the same physical network.
- If the data is already encrypted, this isn't a problem.
- If the data is *not* already encrypted, anyone on the same network as you can see exactly what you're doing online.
- Password-protecting WiFi networks prevents strangers from snooping, but does not prevent other people on the network from reading data.

Google Street View and WiFi

Street View: What Happened

- Google's Street View cars drove around snapping pictures of what was visible from the street.
- To improve location information, Google recorded information about unsecured WiFi networks it drove by:
  - SSID (network name)
  - MAC address (hardware identification)
  - Signal strength (proximity)
- The Street View cars also stored payloads (actual Internet traffic) from unsecured wireless networks; about 200GB of payloads stored.

Why Record WiFi?

- Recording WiFi information around the US makes it possible to get much more precise location information.
  - Searches done from a particular WiFi network can be localized based on the position of the car.
  - Searches done near other WiFi networks can be used to help pinpoint the user.
  - Position-based information helps give better search results and better advertising.
- WiFi payloads are not required to do any of this.

Why Record Packets?

- Recording network packets would give engineers a better view of realistic web traffic.
- For example, could see
  - which websites people were visiting,
  - contents of those websites,
  - distribution of those websites,
  - number of emails sent,
  - contents of emails sent,
  - etc.

Why is this a problem?

- According to Canadian legal documents, the data Google gathered included
  - medical records,
  - website passwords,
  - email exchanges,
  - compromising content,
  - etc.
- Because of the extra location information gathered, this information could be traced back to individual users.

How did Google let this happen?

- Public court documents suggest that a single engineer (“Engineer Doe”) was responsible for installing the packet recording code into the Street View cars.
- How did this happen?
  - Diffusion of responsibility: everyone (probably) assumed that Engineer Doe knew what he/she was doing.
  - Engineer Doe's design documents and presentations did not make impact clear.
  - Incorrect assumptions: engineers believed that only fragments of data would be caught, not full transmissions.
  - Engineers reviewing the code were mostly checking for code syntax and style and did not recognize the significance of storing payloads.
  - Engineers testing the cars did not notice (or did not recognize the significance of) what they were logging.

Why No Encryption?

- Why were there so many unencrypted wireless networks?
- No incentive for WiFi router manufacturers to add encryption by default.
  - Encrypted WiFi is slower than normal WiFi due to the overhead of encryption and decryption.
  - Encrypted WiFi is harder to set up – all computers using the WiFi need the password, and any new devices added to the network need that password as well.
- Little information available to consumers about the risks of unencrypted WiFi networks.

Why No Encryption? Part II

- If communications over the network were encrypted, no personal information could have been logged.
- Why wasn't that information encrypted?
  - End users can't use encryption if servers don't support it.
  - Economic incentives for companies to leave data unencrypted.
    - Encrypted communication is slower than unencrypted communication.
    - Encrypting data increases server expenses and increases response time.
  - Low external pressure on companies to add encryption.
  - Poor communication of risk to average users.

Why Only Unencrypted Networks?

- It is possible to record SSIDs, MAC addresses, and signal strengths of *any* wireless network, even encrypted ones.
- It is only possible to record (intelligble) payloads from unencrypted networks.
- It's unclear whether it is even legal to record data from public WiFi in the first place.
  - This is currently working through the appeals courts.
- I could not find any public documents explaining the engineers' decisions.
- Possible reasons:
  - Concern that encrypted WiFi was intended to be kept private.
  - Concerns about the legality of recording data from private networks.

Why Look at This?

- I chose this example to point out the following questions:
  - How should we design technology in a way that minimizes or properly communicates risks to users?
  - How should we incentivize technology companies to protect privacy and security when, in many cases, only the company itself fully understands the scope of what it's doing?
- These are ethical, societal, and political questions, not technical ones.
- Having a deeper understanding of the technology can help inform your answers.

Topical Announcement

- New joint majors announced:
  CS + English and CS + Music.
  - Complete both degrees.
  - Each department waives two requirements.
  - Complete a joint capstone project at the end.
- Want to learn more? Come talk to me after lecture!