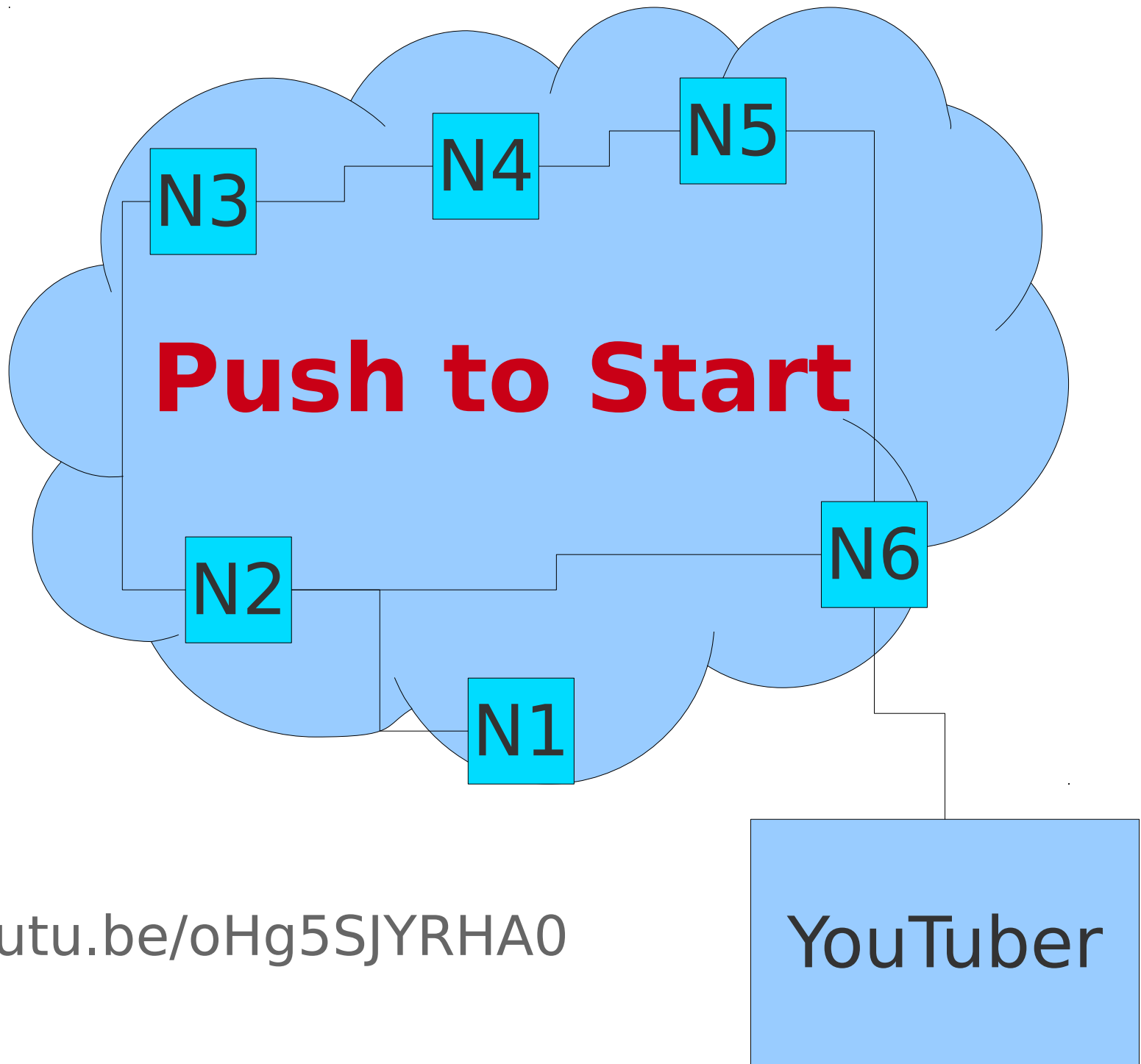


# How You're Going to Rule the World

Wolf Richter

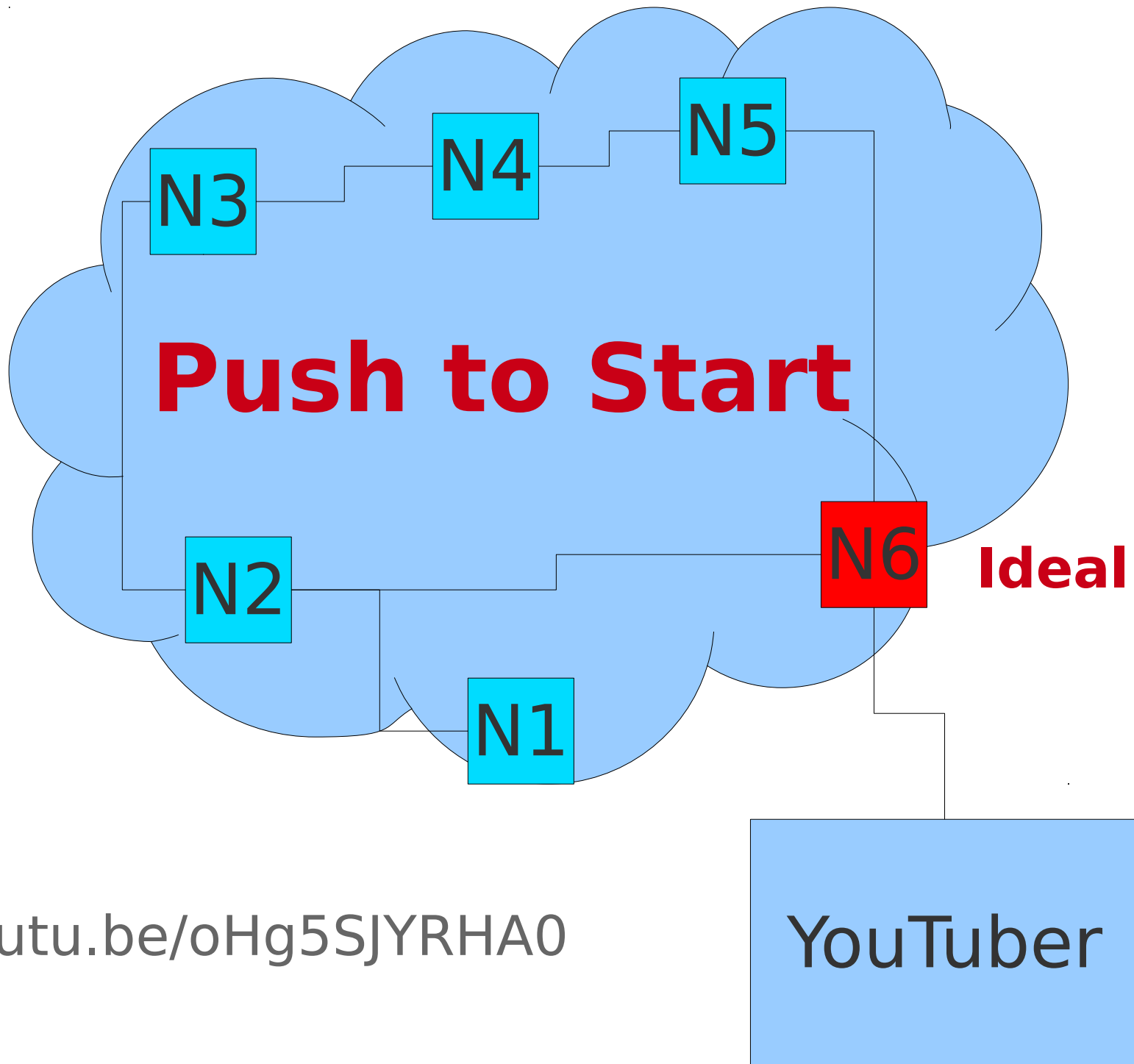
# Content Distribution Network

- Global
- Increase access bandwidth to objects
- Decrease latency
- Akamai, Limelight, Google, Facebook, ...
- Often use DNS to route clients
- Route you to the nearest front-end node



**I want:**

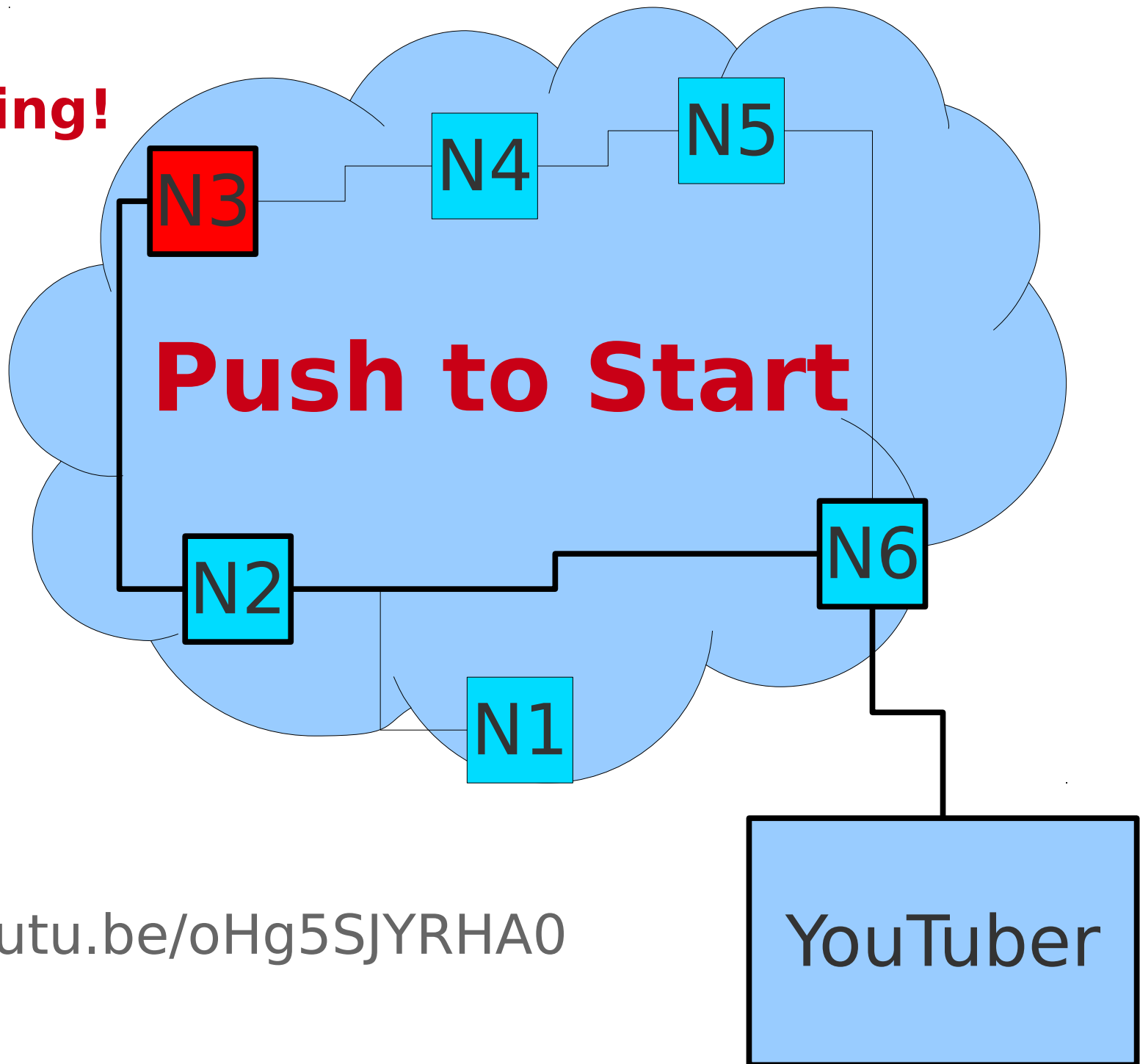
<http://youtu.be/oHg5SJYRHA0>



**I want:**

<http://youtu.be/oHg5SJYRHA0>

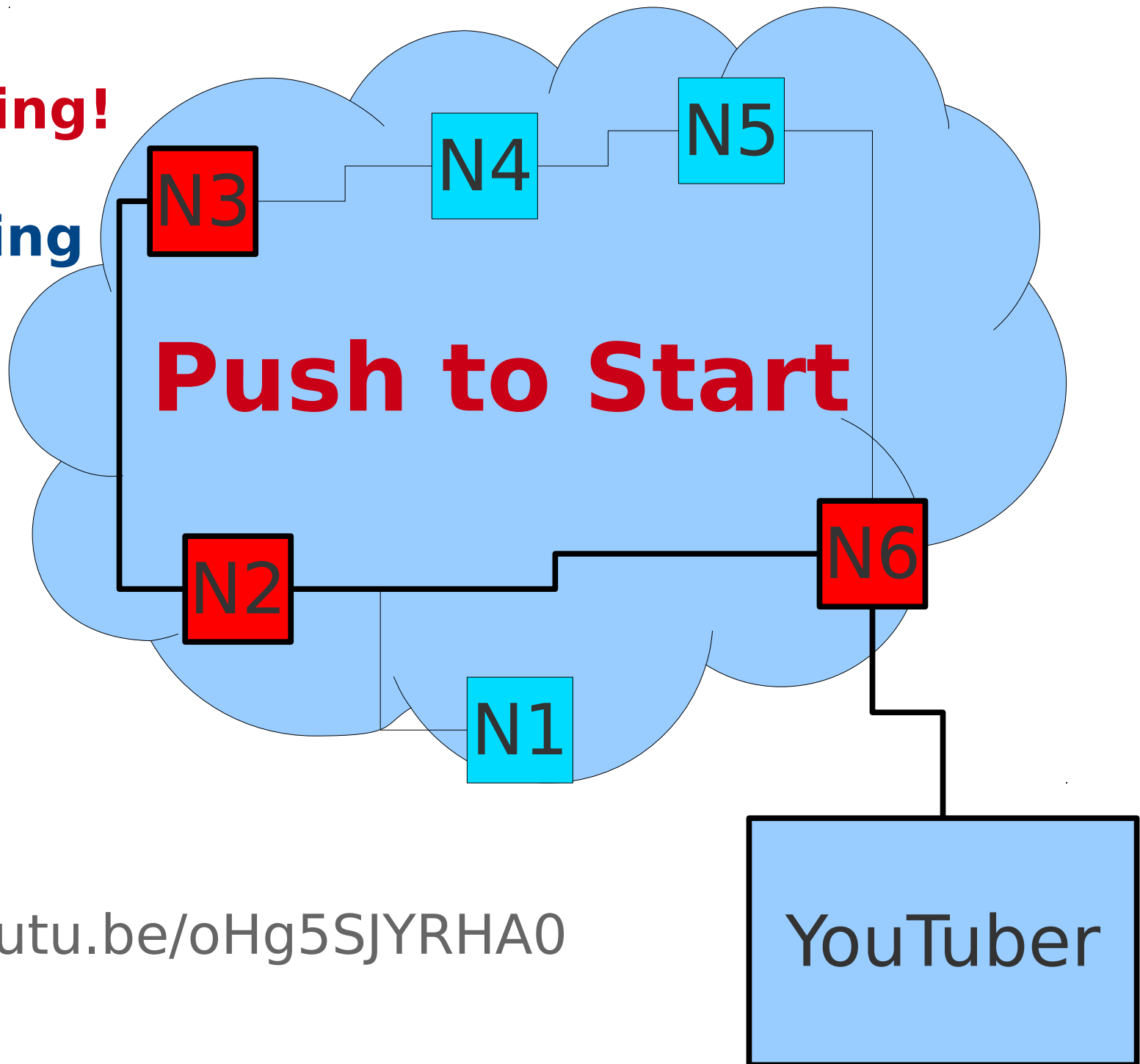
**Routing!**



**I want:**

<http://youtu.be/oHg5SJYRHA0>

**Routing!**  
**With**  
**caching**



**I want:**  
<http://youtu.be/oHg5SJYRHA0>

That's important, say, when  
Koreans from around the  
world want to listen to

Girl's Generation

# Applications!?

- CDN
- **letscrate.com** – recent startup
  - Basically, your web app with a pro design
  - Literally **you add files**, and **then you get them**
- Amazon S3
- Dropbox
- YouTube
- Flickr



# What are you Building?

- Framework for **content discovery**
- and **content distribution**
- OSPF routing daemons provide both
  - Discovery – **reliable flooding** mechanism
  - Distribution – **addfile** intelligently, or **cache...**
- You are **building a framework**
- Fully **internal routing**, designated front-ends
- Route data to clients

# Routing Daemons [CP2]

- Communicate via UDP
- Example UDP code uploaded
- Implement OSPF reliable flooding
- Return URLs to clients
  - Routing to another peer  
[http://peername:peerport/rd/rdport/object\\_name](http://peername:peerport/rd/rdport/object_name)
  - Or locally  
<http://localhost:port/static/sha256sum>

# ~~Open Shortest Path First~~

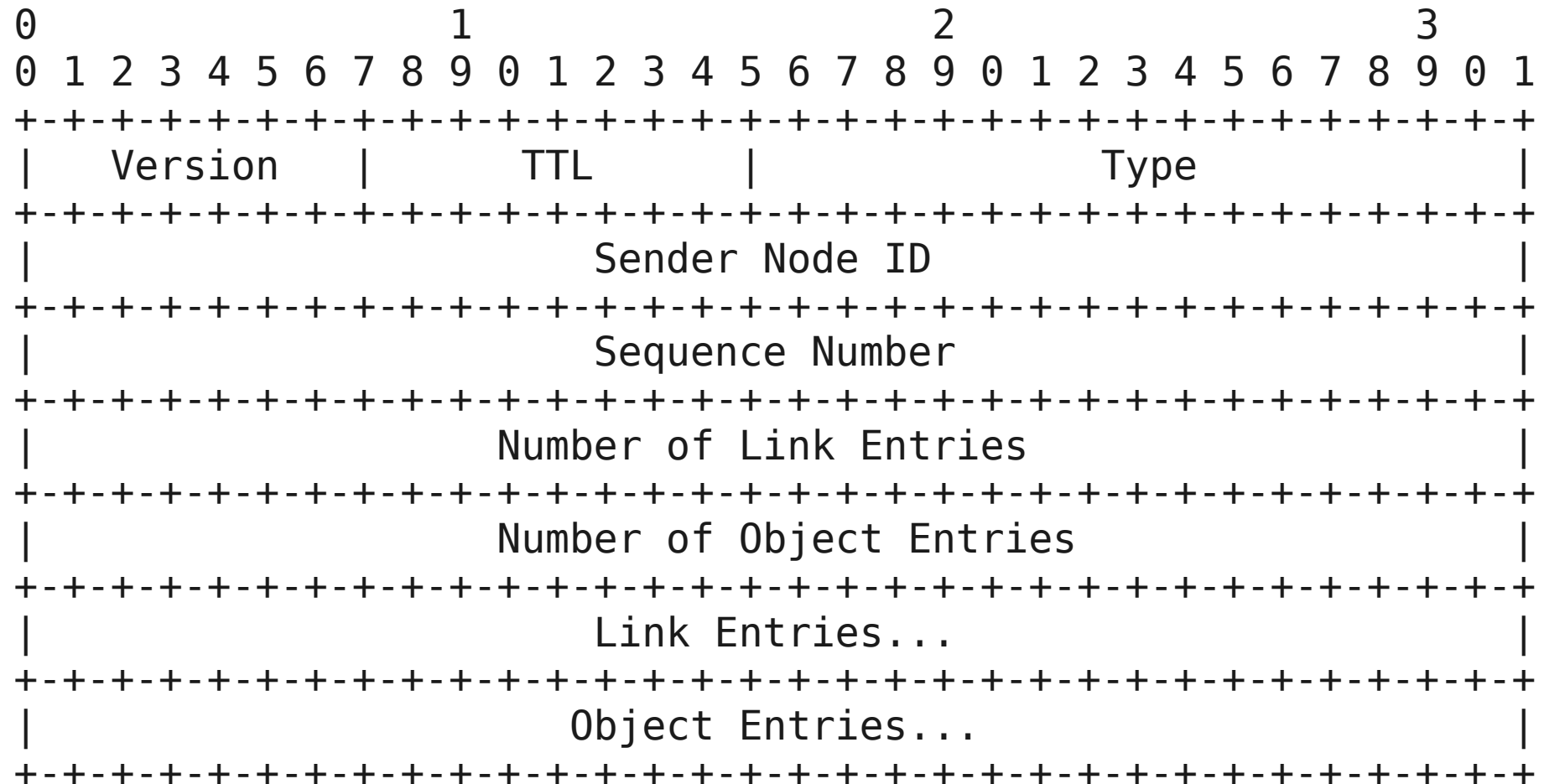
- But, really, only **bastardized LSAs**
- Implement the sending/receiving exactly
- **LSA Packet**
- **LSA ACK Packet**
- **Network byte order!**
- **Triggered Updates: node down, new object**
- Respect TTLs according to specification
- TTL = **0** → Machine offline, delete entries

# RD == LSA Hub

- LSAs come in, **forward** to other neighbors
- **Learn** a lot from them
  - Graph of network
  - Objects on nodes
- Oh isn't this easier than Project 1!?
- **Data structures and algorithms...**

# Link State Advertisement (LSA)

- Secret: ads make money on the Internet
- But, not quite these kind...
- Update frequency: 30 seconds



# Default Values

- Version = 0x01
- TTL = 0x020
- Type = 0x0
- Sequence Number = 0
  - Monotonically increasing
- Number of Link Entries = 1 (receiver)
- Number of Object Entries = 0

# Entries Lists

- **Two byte length** in number of items
  - `uint16_t`
- Then entries
- **4 bytes per link entry** (node id's)
  - `uint32_t`
- **Object entries** are arbitrary length
  - Each **preceded by 4 byte length**
  - `uint32_t`

# In Bytes...

- Links

[4 byte node1][4 byte node2]...

- Objects

[4 byte len1][...][4 byte len2][...]...



# LSA ACK

- Version = 0x01
- TTL = 0x0
- Type = 0x01
- Sequence Number = same as received
- Number of Link Entries = 0
- Number of Object Entries = 0

# What does RD Track?

- **Graph of network** → to find shortest path
- **Object names** → next node, port, sha256
- Node → objects
- Node → neighbors
- Storing this data is **up to you**
- **Think hard with your partner**
- **This is 60 points**

# Rules?

- Always route via shortest path
- Object retrieval looks like:
  - front-end → send\_file(urlopen(x1))

# Rules?

- Always route via shortest path
- Object retrieval looks like:
  - front-end → send\_file(urlopen(x1))
  - x1 → send\_file(urlopen(x2))

# Rules?

- Always route via shortest path
- Object retrieval looks like:
  - front-end → send\_file(urlopen(x1))
  - x1 → send\_file(urlopen(x2))
  - x2 → send\_file(urlopen(x3))

# Rules?

- Always route via shortest path
- Object retrieval looks like:
  - front-end → send\_file(urlopen(x1))
  - x1 → send\_file(urlopen(x2))
  - x2 → send\_file(urlopen(x3))
  - x3 → send\_file(urlopen(x3-static))

# Rules?

- Always route via shortest path
- Object retrieval looks like:
  - front-end → send\_file(urlopen(x1))
  - x1 → send\_file(urlopen(x2))
  - x2 → send\_file(urlopen(x3))
  - x3 → send\_file(urlopen(x3-static))
  - x3 send\_file ← Flask static server

# Rules?

- Always route via shortest path
- Object retrieval looks like:
  - front-end → send\_file(urlopen(x1))
  - x1 → send\_file(urlopen(x2))
  - x2 → send\_file(urlopen(x3))
  - x3 → send\_file(urlopen(x3-static))
  - x3 send\_file ← Flask static server
  - x2 send\_file ← x3 send\_file



# Rules?

- Always route via shortest path
- Object retrieval looks like:
  - front-end → send\_file(urlopen(x1))
  - x1 → send\_file(urlopen(x2))
  - x2 → send\_file(urlopen(x3))
  - x3 → send\_file(urlopen(x3-static))
  - x3 send\_file ← Flask static server
  - x2 send\_file ← x3 send\_file
  - x1 send\_file ← x2 send\_file

# Rules?

- Always route via shortest path
- Object retrieval looks like:
  - front-end → send\_file(urlopen(x1))
  - x1 → send\_file(urlopen(x2))
  - x2 → send\_file(urlopen(x3))
  - x3 → send\_file(urlopen(x3-static))
  - x3 send\_file ← Flask static server
  - x2 send\_file ← x3 send\_file
  - x1 send\_file ← x2 send\_file
  - front-end ← x1 send\_file

# Extra Credit Before: LPTHW

- LPTHW – By November 19
  - Email Wolf
  - What you did, where it is, short report (1-2 paragraphs)
  - How effective was it, how long it takes, what it's missing

# Extra Credit PJ2: Caching [10]

- Optional **final command-line argument**
  - [cache size in bytes] – default 1 Gibibyte
- **Choose a caching policy** (LRU etc.)
- **Save objects fetched remotely** in /static/
- Name them according to **sha256sum**
- **ADDFILE** for this object to yourself

# Extra Credit PJ2: Longest Prefix [10]

- Assume **object names have structure**
- Based on **'/'** (**'/cmu/csd/'**)
- **Match based on longest prefix matching**
- **Nodes have** {**1** : **'/cmu/'**, **2** : **'/cmu/csd/'**}
- **Want:** Nodes[**'/cmu/csd/srini'**] → **2**
- Regular **'files'** can be mixed with **'directories'**

# Extra Credit PJ2: Pipelining [10]

- Modify Flask to maintain **10 connections**
- Global pool
- These **10** service all RD requests
- Perhaps consider: **Python Synchronized Queue**
  - Get connections off for use
  - Put them back when response received

# What's the Deal?

- Choose 2 from the 3 EC's presented
- Turn them in at final submission time
- [0,10,20] points EC on PJ2
- [0,10] points EC from LPTHW
- 30 points EC possible at this point

# PJ1 Leaderboard

adityaa1  
alussier  
anandsur  
angx  
apodolsk  
brclark  
bstrassm  
chunhowt  
dcrescim  
ebreder

gyang1  
hanl1  
hongjaic  
huiyangx  
jchee  
jcmacdon  
jwloh  
kailili  
kbaysal  
kdalmia

mdan  
mengh  
mfurman  
minjaele  
mkahn  
moz  
mswang  
mteh  
nsegall  
ochoe

phoskins  
rggonzal  
seunghwl  
siyounggo  
sjoo  
spradhan  
syedkar  
tbach  
tbenshac  
tianyec

weishi  
xuanzhan  
yueyuan  
zhuojil  
ziccardi



# PJ1 Regrades

11AM – 3PM Saturday

GHC 9127

GitHub:

Git it, got it, good.

```
git clone git://github.com/theonewolf/15-441-Recitation-Sessions.git
```