Routing and route finding in unstructured networks

Andres Erbsen Sandra Schumann

December 5, 2012

Who on earth are we anyway?

Andres Erbsen Sandra Schumann

Tallinn Secondary Science School, 12th grade

We like Haskell, Python, C++, Linux, the Unix way

Email prioritization
Document processing:
LaTeX, markdown

Statistics
Teaching:
Python and physics

We run a seminar at school

Online courses from around the world:
AI. Algorithms. Crypto. Information Assurance. Robotics.

It gets tiresome being spoken to as if you are a child, even if you happen to be one. – the good thing about the Internet

Motivation

Networks?

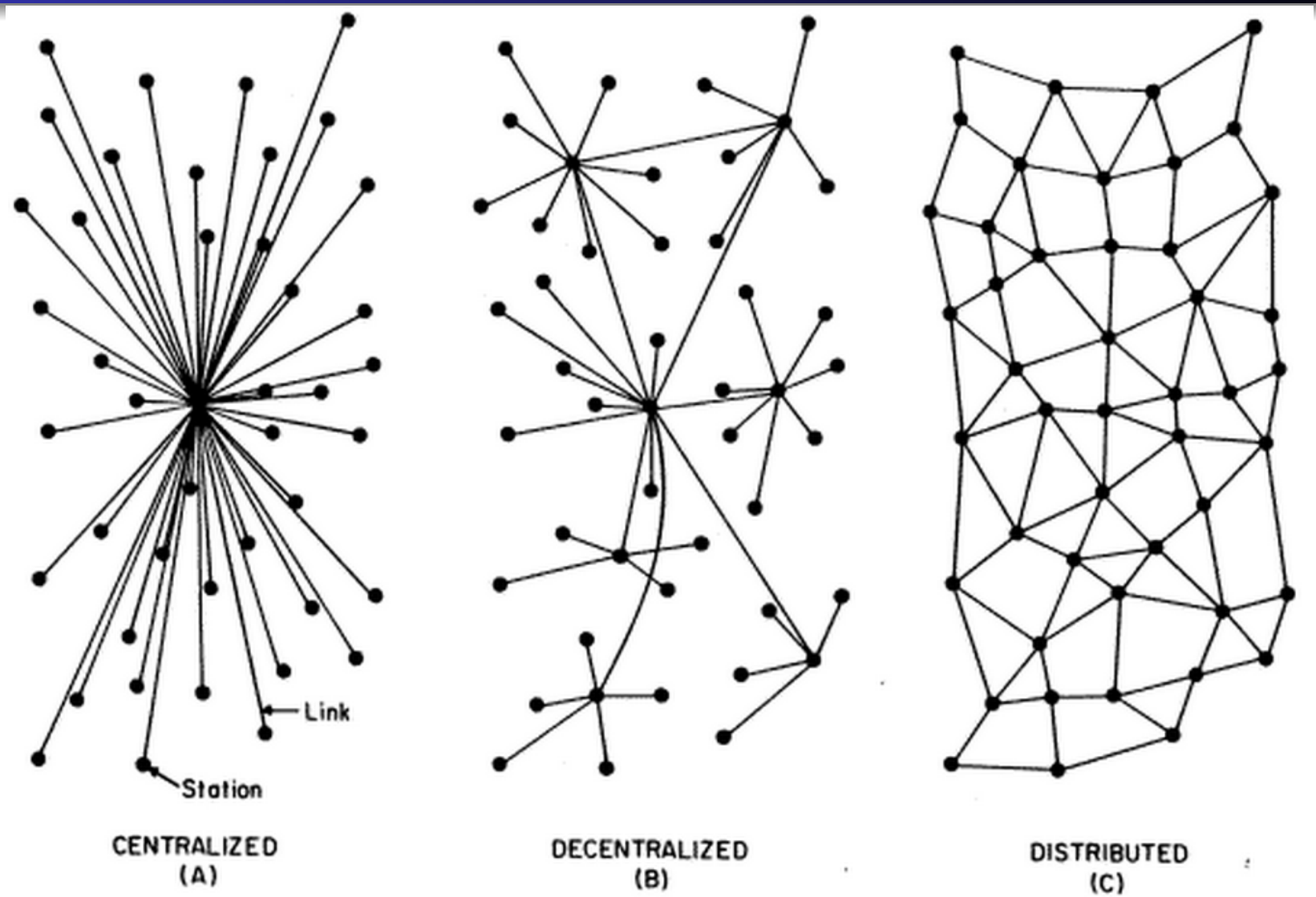


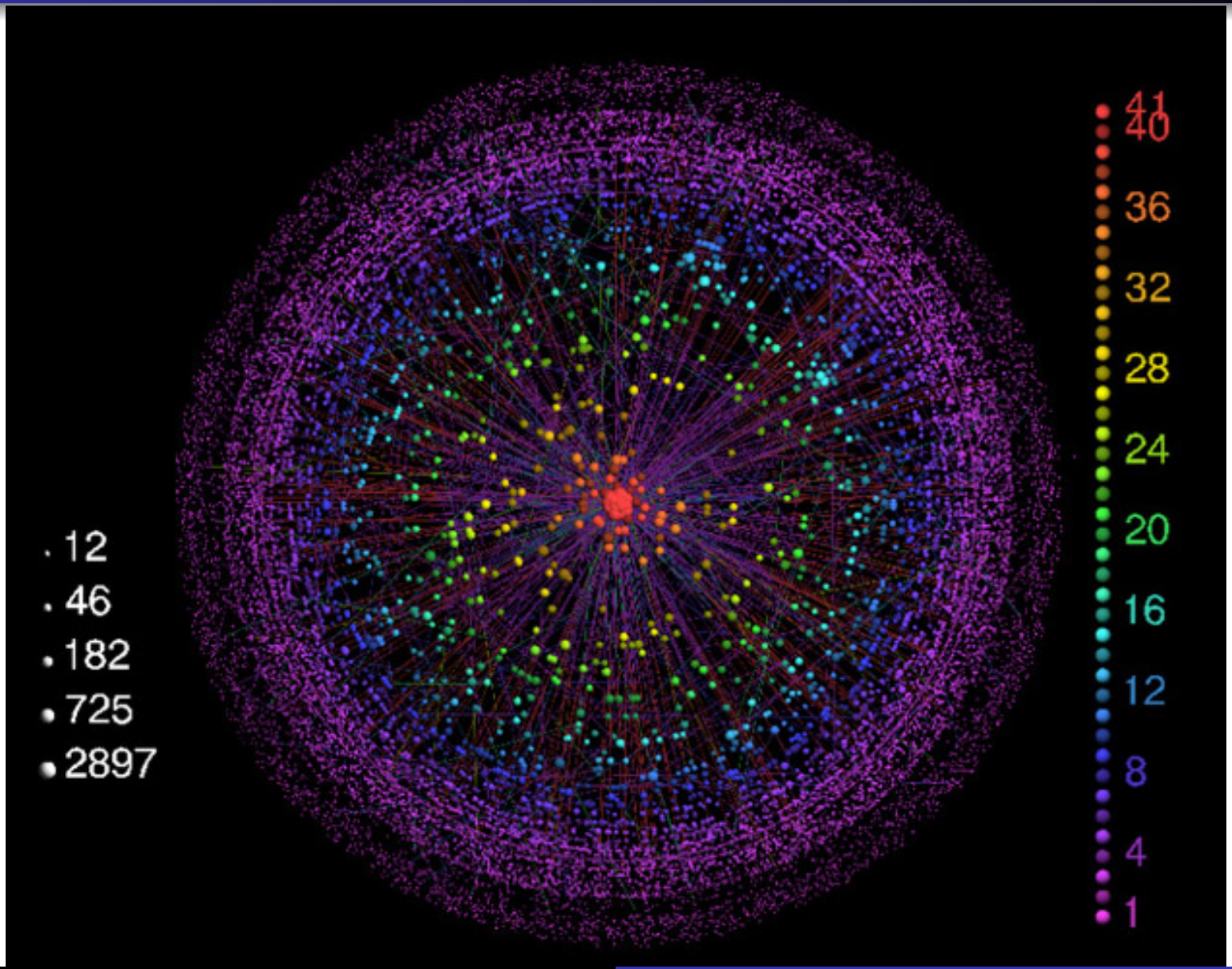
FIG. 1 — Centralized, Decentralized and Distributed Networks

Why?

Since destruction of a small number of nodes in a decentralized network can destroy communications, the properties, problems, and hopes of building "distributed" communications networks are of paramount interest.

– P. Baran, 1964. On Distributed Communications.

The Internet: between ISPs



The Internet: the problems appear

- Internet protocol suite: proactive routing by address prefix
- 42K Autonomous Systems, BGP routing
- An AS has many subordinates: $\frac{2^{32}}{42000} \approx 100000$
- Inter-AS links must be in BGP tables
- The BGP subgraph must be connected
- Everyone is reached through exactly one AS
- ...
- (Normal) routers need to be told “the Internet is this way”
- Most end-user devices are only capable of being leafs
- Lots of hacks and workarounds

What has changed?
This was already solved once!

- The internet design *is* based on good resarch
- In 1964 there were mere *hopes* of building distributed networks

- Diffie–Hellman–Merkle 1976: public-key cryptography
- Fundamental change: all devices in a network do **not** have to trust each other ultimately to be mutually useful

Political motivation

If we are going to build systems of communication for future politics, we're going to have to build them under the assumption that the network is not only untrusted, but untrustworthy. We're going to have to build under the assumption that centralized services can kill you.

Well, we can go back to mesh networking. We've got to go back to mesh networking.

Because we have friends in the street trying to create human freedom, and if we don't help them, they'll get hurt. We rise to challenges, this is one.

Eben Moglen (founder of SFLC, FreedomBox) at FOSDEM 2011

Goals

Goals

- Resiliency
- Privacy (security)
- Scalability

Resiliency

- Any device can connect to **any** other device*
  - No NAT, no "port forwarding" needed
  - Any user can run a public service (e.g. instant messaging)
- No built-in restrictions on the network structure
  - Making use of redundancy is hard
- Automatisation - doable
  - Manual (mis-)configuration is a common issue
  - Human intervention: slow, expensive

*as long as there exists a physical path

Privacy of the transferred data

Confidentiality against espionage

When sending packet to someone, **only** they can read it.*
* Disclaimer: does not protect against untrustworthy recipients.

Integrity against corruption and authentication forgery

- Packet came from who it says it came from.
- Packet has not been modified on the way.

Repudiability

The recipient cannot convince *anyone else* that the packet came from the sender (could have just made it up).

Forward secrecy - requires another round trip at initialization

Privacy of activity and identity

- Efficiency tradeoff, thus optional
- Quite like TOR, but not bloat – too easy not to do

Possible to hide from non-global adversaries

- Who is that device connecting to?
- What the route is?
- Whose packets am I forwarding? To whom?
- Who just connected to me? From where?
- Are A and B talking to each other?

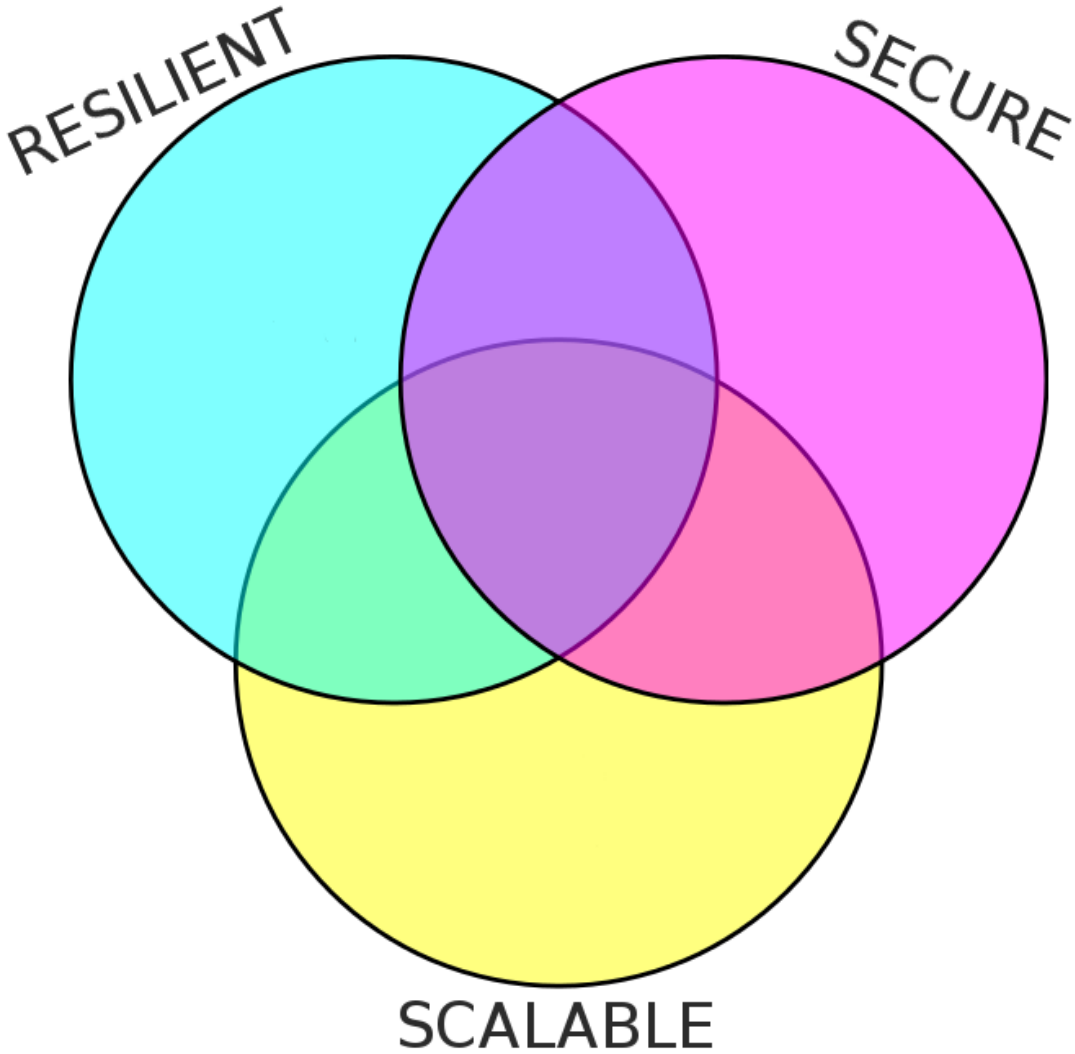
Scalability

Keep enabling good connections as the network grows

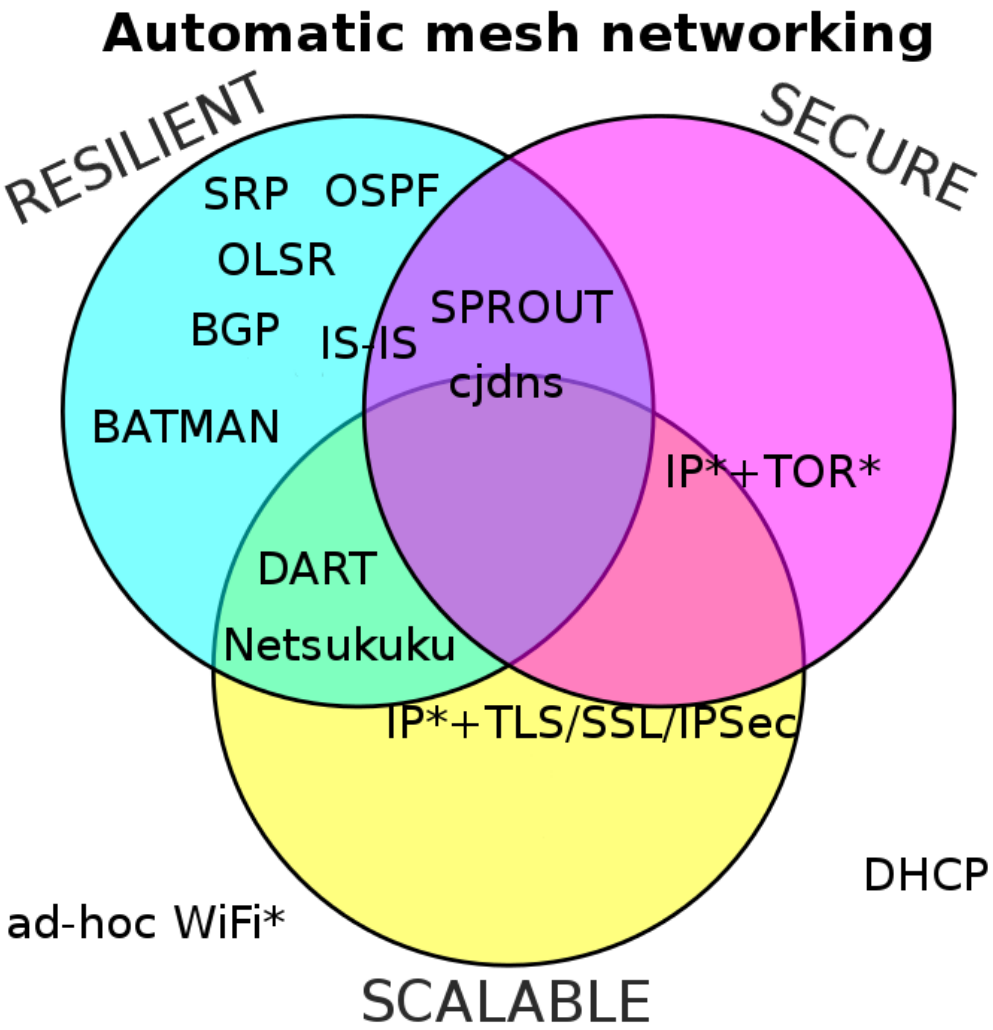
- Most mesh network protocols are $\Theta(n^2)$
- Without sacrificing privacy (unlike DART and Netsukuku)

Billions of nodes - challenging

Three goals



Three goals


Automatic mesh networking
RESILIENT
SECURE
SRP
OSPF
OLSR
BGP
IS-IS
SPROUT
cjdns
BATMAN
IP*+TOR*
DART
Netsukuku
IP*+TLS/SSL/IPSec
DHCP
ad-hoc WiFi*
SCALABLE
*needs manual configuration

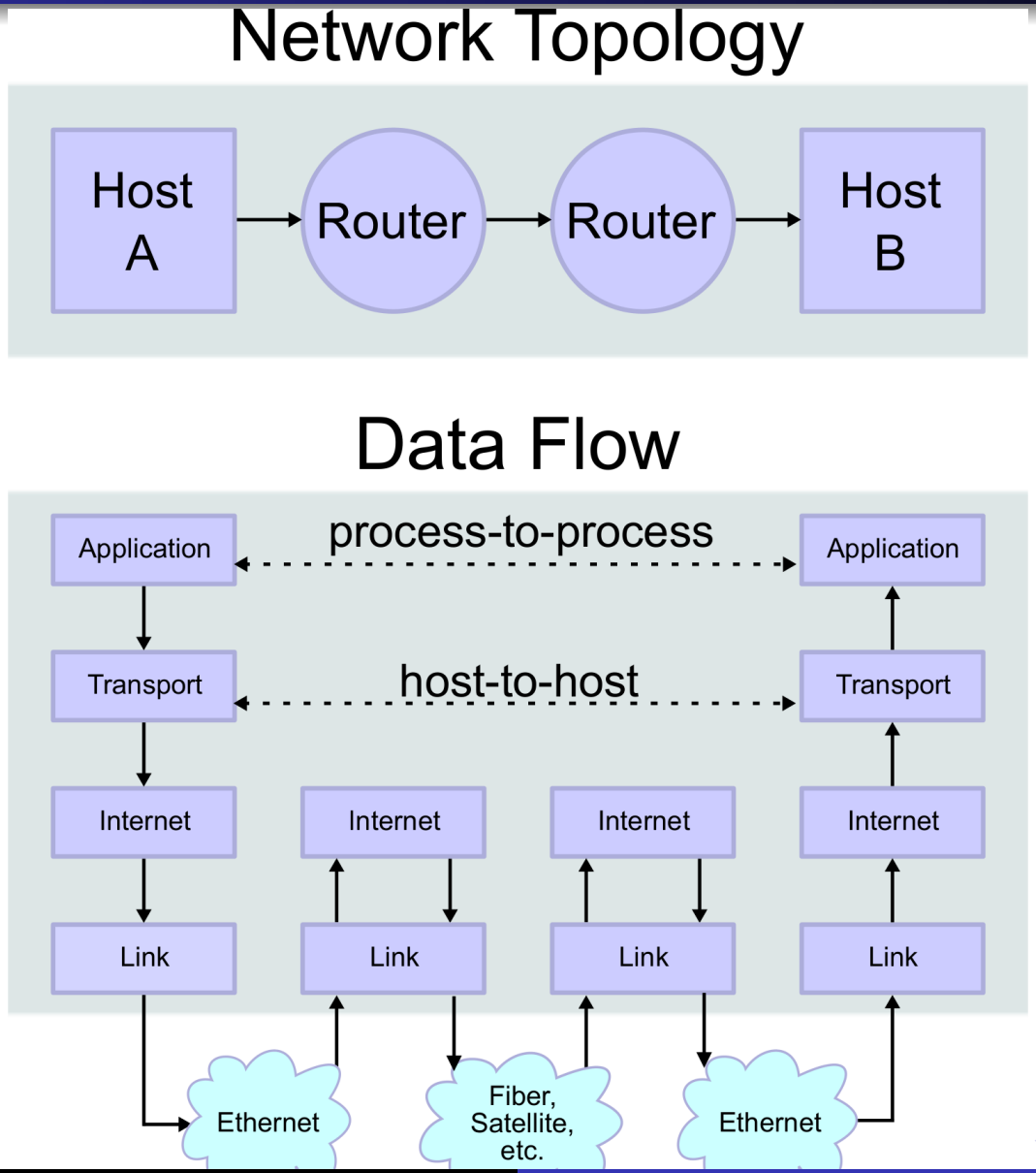
Summary

A good system...

- is more **resilient** than an IP-based one
- supports **privacy** comparable to CurveCP and TOR combined
- **scales** better than cjdns (with hope to improve further)

Means

A new network ("internet") layer protocol



Extensible and non-uniform

- Different compromises between privacy and efficiency
  - many servers are perfectly fine with publishing their location
  - the same device may need a fast connection and an anonymous one
  - anonymizing forwarders are useful but rare
- The client software *won't* be updated on time, tolerate that
  - backwards compatability
  - algorithm agility
- Route finding is hard, there are many ways to do it
  - allow for different algorithms to be used in the same network
  - separate route finding from routing
  - make routes verifiable – no more arp spoofing!

The Basics

- Devices: computers, cellphones, routers, fridges, VM-s...
- Links: ethernet cables, wifi networks, pigeons, fiber...
- A device - anything in the network that
  - has an unambigous and verifiable identity
- A link - a promise to forward packets
  - nobody can force them
  - if we trust them, we know that the link works...
  - ...otherwise just try, it still might
  - digitally signed to prevent forgeries

A device

- Known by it's public signing key (hash of it)
- Can and will be made to prove its identity
  - decrypt something encrypted with their public key
  - sign a challenge?
- Other keys must be signed with the main key (like PGP)
- No CA-s needed at this level
- Web of trust is recommended

Links

- Signed by both ends (like SPROUT)
- Terms of service included
  - round trip time, minimal transmission unit
  - "only for X"
- Timestamp!
- Links that lead from a device to another make a *route*
  - anyone can compose routes
  - digital signatures are still valid
  - most link metrics can be combined into a route metric
- Even more links $\rightarrow$ graph
  - shortest paths in a local subgraph of the network (like OSPF)
  - extra: shortest disjoint paths for multiple connections

Route finding vs routing

Separate route finding from routing

Routing

The node who starts a connection specifies everything step by step and hop by hop. We don't think there is another way.

- Before any data is transferred, the connection is set up explicitly.
  - every node stores the next hop
  - ... and the previous hop to prevent routing loops
  - negotiate cryptograpic algorithms and routing procedures
- Each packet needs just "connection id" field
- No source address, no destination address, no "options", no fragment info...

Types of connections

1. Private communication
   - most of the traffic!
   - encrypted, authenticated
   - not that anonymous
     - but still no world-readable address on the envelope
2. Best-effort anonymity
   - protecting against comperhensive traffic analysis is very hard
   - hard to distinguish from 1.
   - onion-encryption (like TOR)
   - throwing in intentional delays is easy
3. Public large data
   - trade privacy for optimizations
   - the privacy-oriented scheme is (like IP) costly to implement in hardware

Flexibility

- All of these types can be mixed in the same connection
- Per hop connection negotiation
- The forwarders can and should be dumb

What can an adversary do?

- Sniffing?
- Man-in-the-middle attacks?
- Misrouting packets?

The worst one can do: drop packets they promised to forward?

Trust – let's rely on experiences!

They forwarded my packet 100 times – probably will do it again.
It hasn't worked 10 times in a row – let's consider another route...
E.g., Bayesian statistics (with Laplace smoothing) could work.

Route finding

- Difficult - there is always room for improvement
- Pluggable!

Using arbitrary routing scheme for route finding

- Information stored by devices remains unchanged
- Instead of routing useful packets, traceroute
  - save the descriptions of the traversed links
  - get the result back to the sender

A sane starting point

- Store one's neighbourhood
- Use Dijkstra's algorithm to find local routes
- Store 'good enough' routes to some other nodes?

- Useful for building other schemes

- Notify other devices of changed links
  - include information about the endpoints
- Forward interesting notifications
  - what's interesting to you, is probably also to your neighbours

Existing solutions

- IP - prebuilt tree structure
- cjdns - random, always walk closer in address space
- Netsukuku and DART - recursively divide into groups

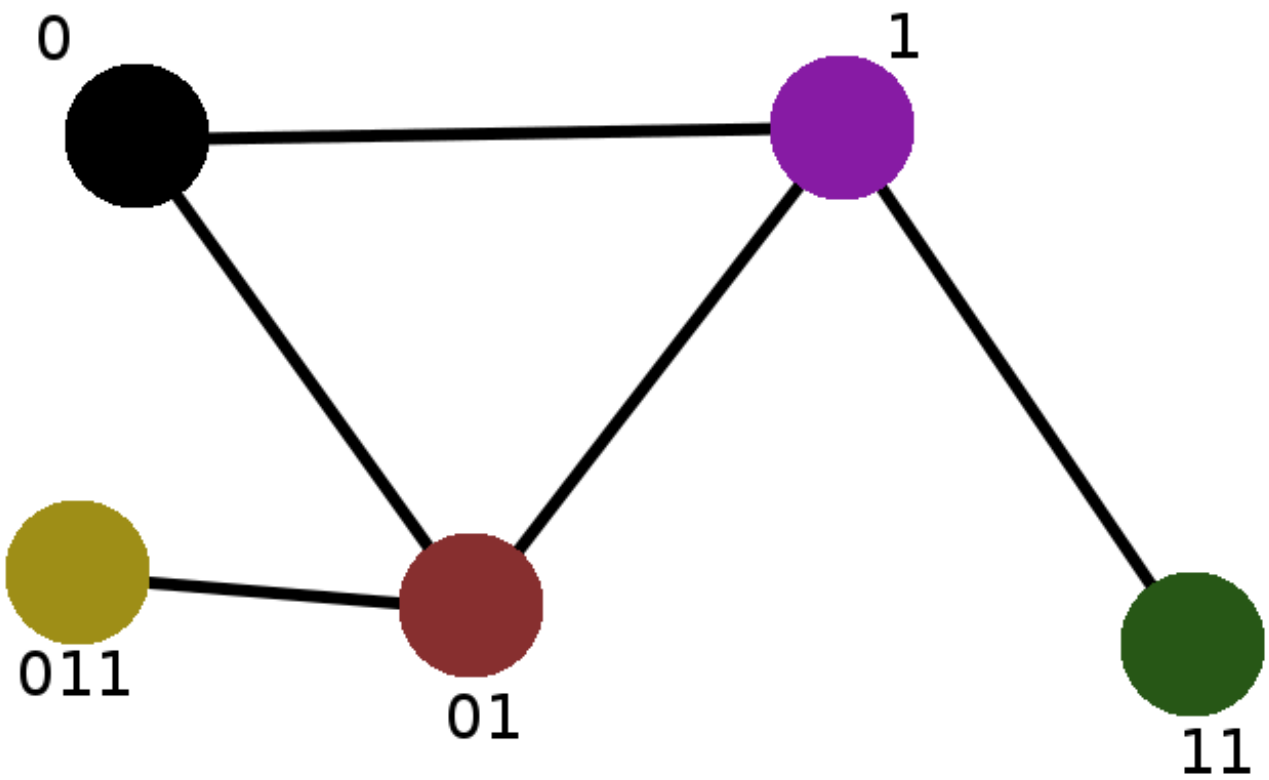
Our solution

- Every device has an “address”
- Sequences of 0s and 1s

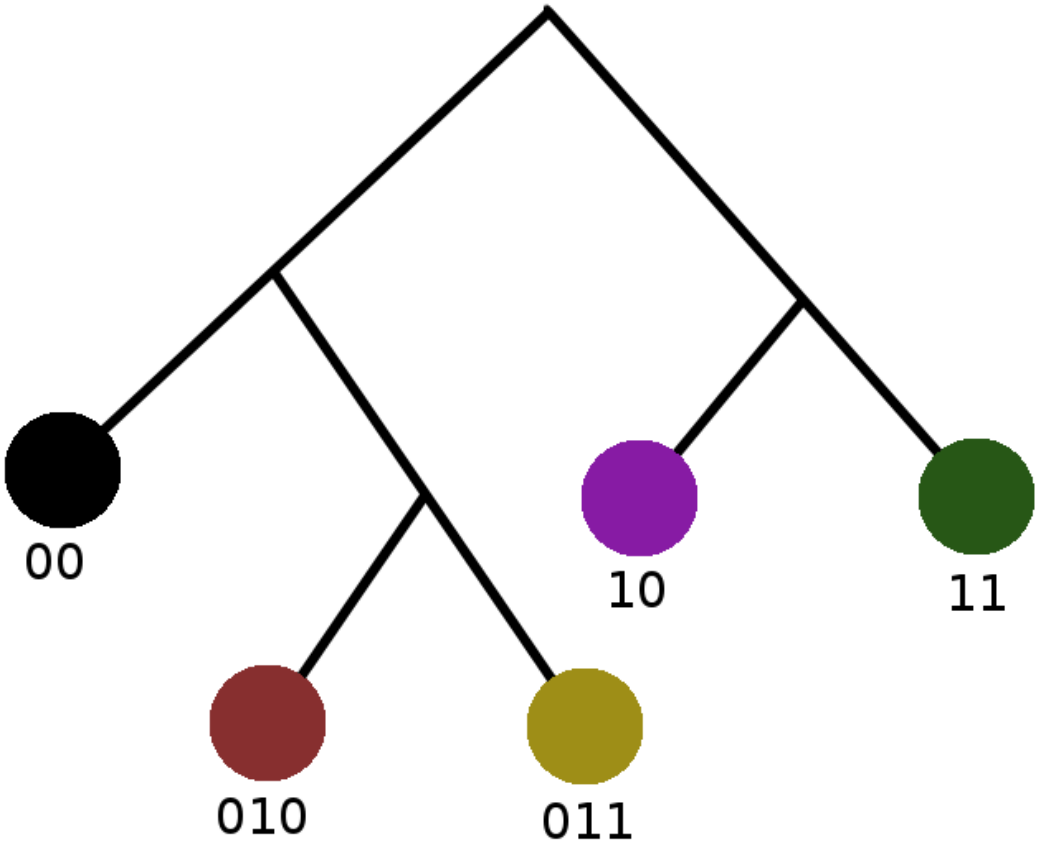
Example

$1000101 = 1000101000 = 100010100000...$

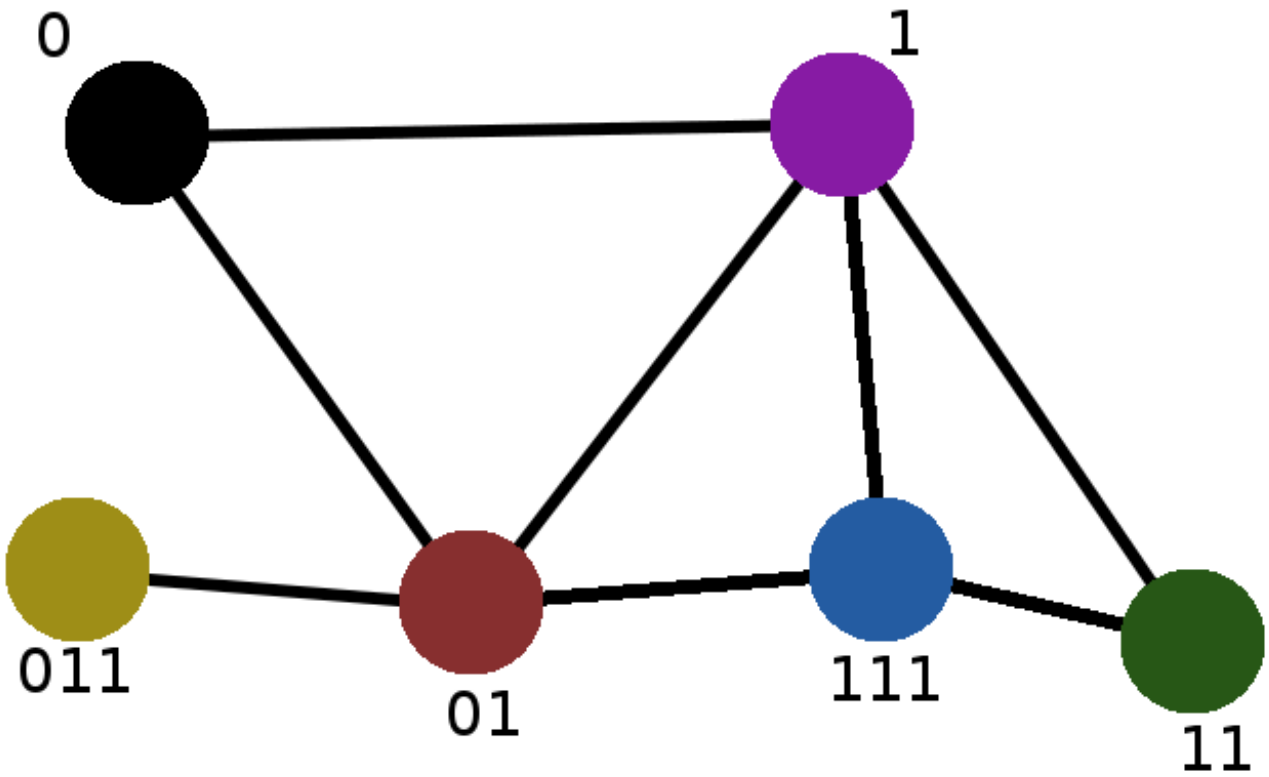
Example of joining the network



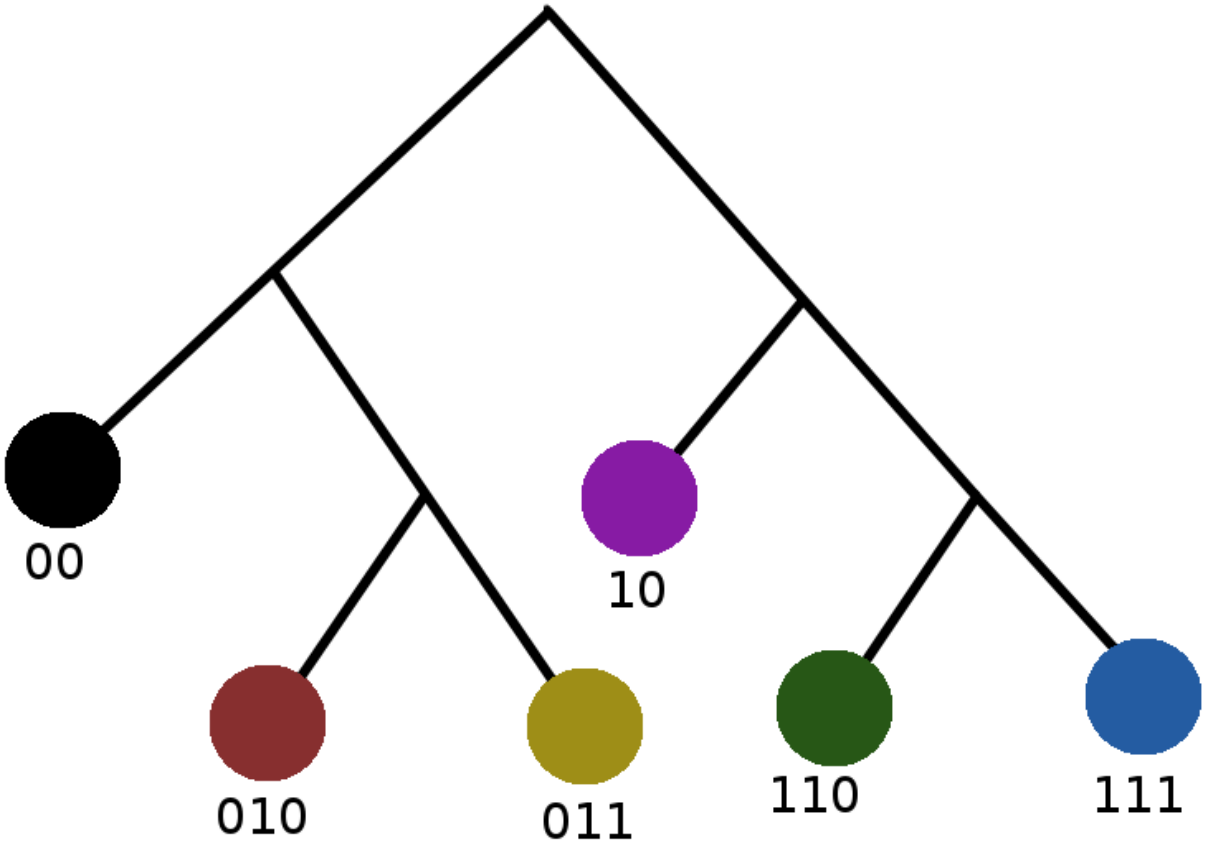
Example of joining the network



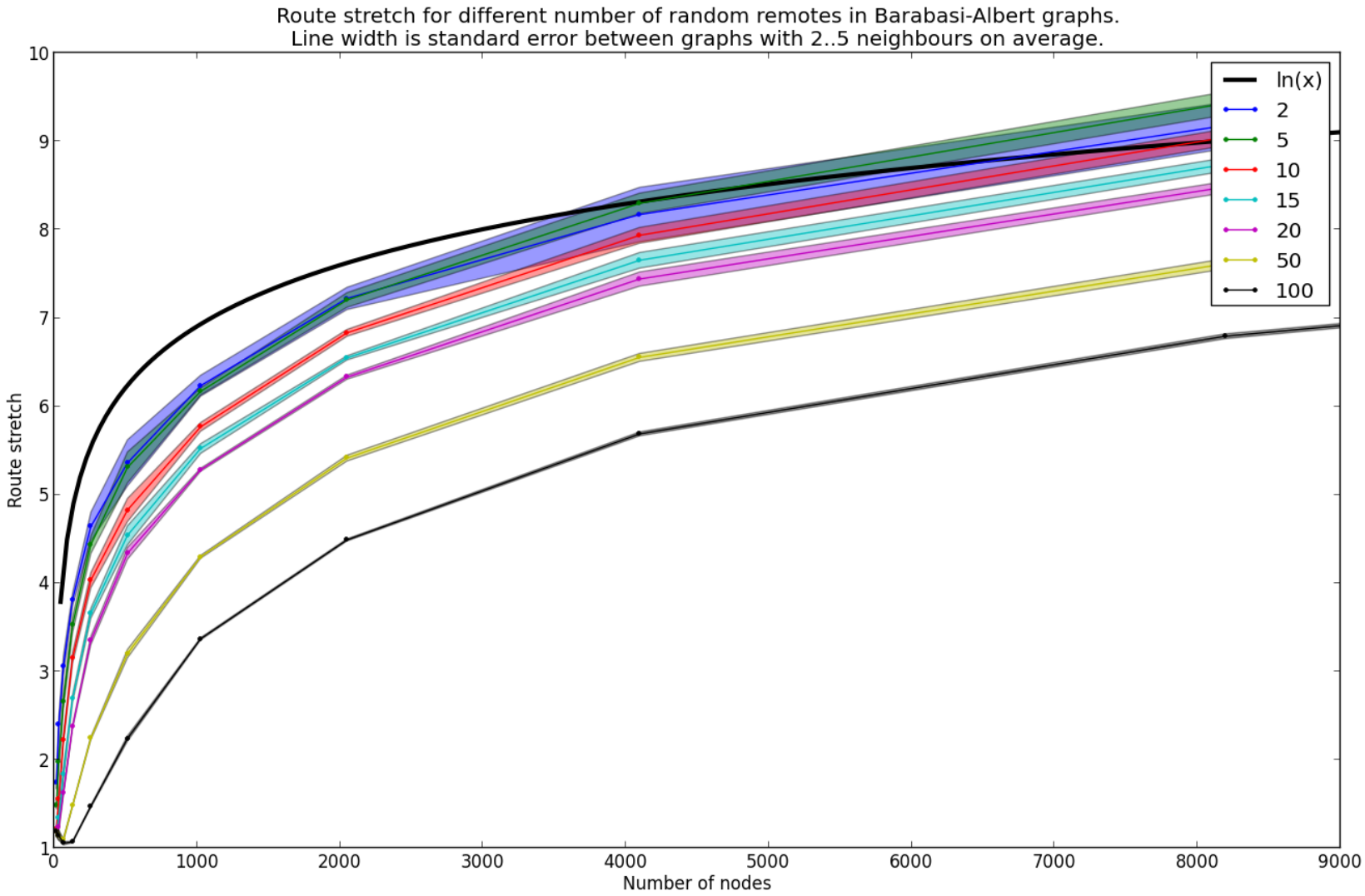
Example of joining the network



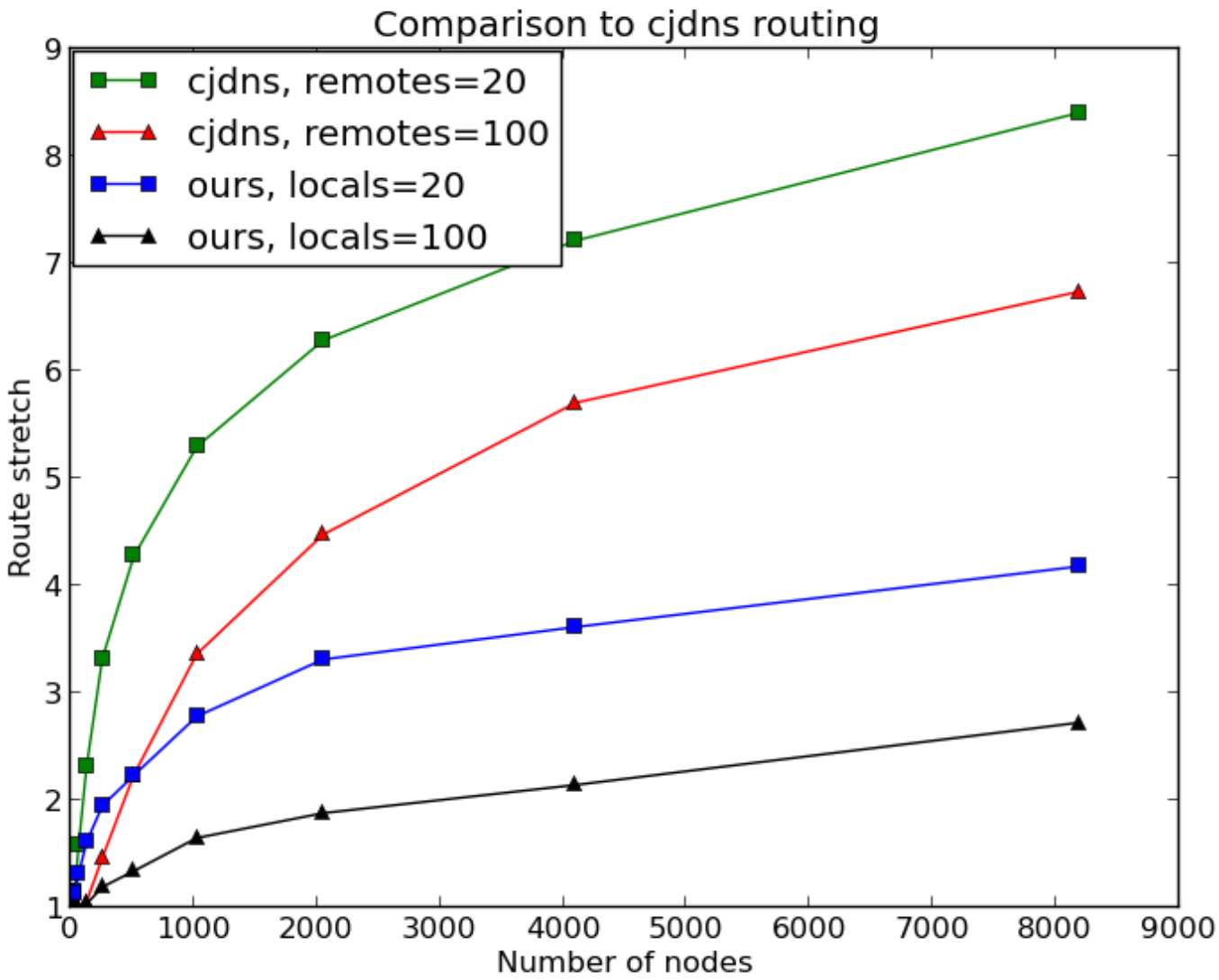
Example of joining the network



Simulating cjdns



Comparison


Comparison to cjdns routing
cjdns, remotes=20
cjdns, remotes=100
ours, locals=20
ours, locals=100
Route stretch
Number of nodes
1
2
3
4
5
6
7
8
9
0
1000
2000
3000
4000
5000
6000
7000
8000
9000

What is your address?

- Devices' addresses may change often
  - This should not cause loss of connectivity
- DHT for identifier $\rightarrow$ address lookups
- Address assignment must be signed – no ambiguity
- Organising DHT by addresses instead of random ID-s decreases complexity

Problems and plans

- Removing nodes works almost like adding them...
- Joining two networks is a mess
- Networks changing quickly cause flood
- Keep address allocation uniform to avoid long addresses

- Implement the whole thing

Backwards compatability

How to get this going...

- Seamlessly interconnect this kind of networks over the Internet
  - easy
  - like any other link
- Connect back to internet over this
  - needs thought, but doable
  - Netsukuku has a design for it
  - just run OpenVPN over it