CLOAK
Virtual Networking Architecture

Damien Magoni

LaBRI – University of Bordeaux

Aims

- What: virtual networking over the Internet
- Which is for:
  - messaging, conferencing, sharing, streaming, socializing
- Why:
  - solve issues (scalability, reliability, simplicity)
  - add features (mobility, flexibility, security, autonomy)

Means

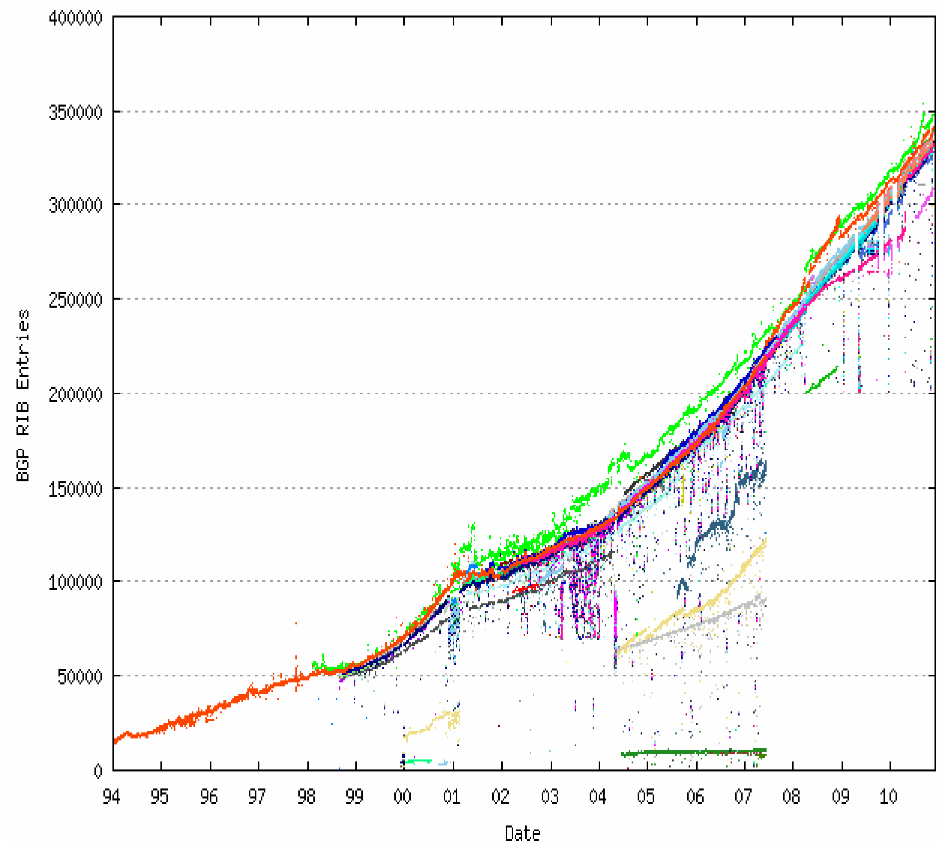
- How:
  - P2P links
  - Overlays
  - Abstract names
  - Distributed Hash Tables
- Where: worldwide Internet
- When: asap!

Current issues in the Internet

- Not scalable
  - Addressing space exhaustion for IPv4
  - Routing tables explosion
- Not reliable (in dynamic environments)
  - Limited mobility at network and transport layer
  - No flexibility (switchable connections)
  - No simultaneous mobility/flexibility + security
- Not simple (broken E2E model)
  - Many addressing spaces (v4/v6/NAT)
  - Many middleboxes (NATs, proxies, etc)
  - No user definable namespaces (only official DNS)

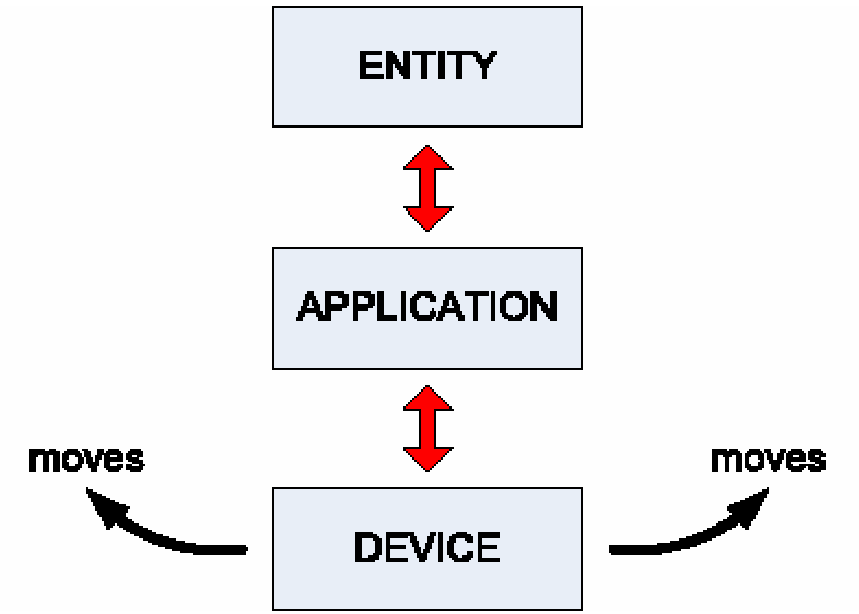
Scalability issues

- IPv4 addressing space exhaustion
- Routing tables explosion (e.g. BGP)
- Current routing protocols are not scalable (OSPF, BGP, AODV, DSR, etc)



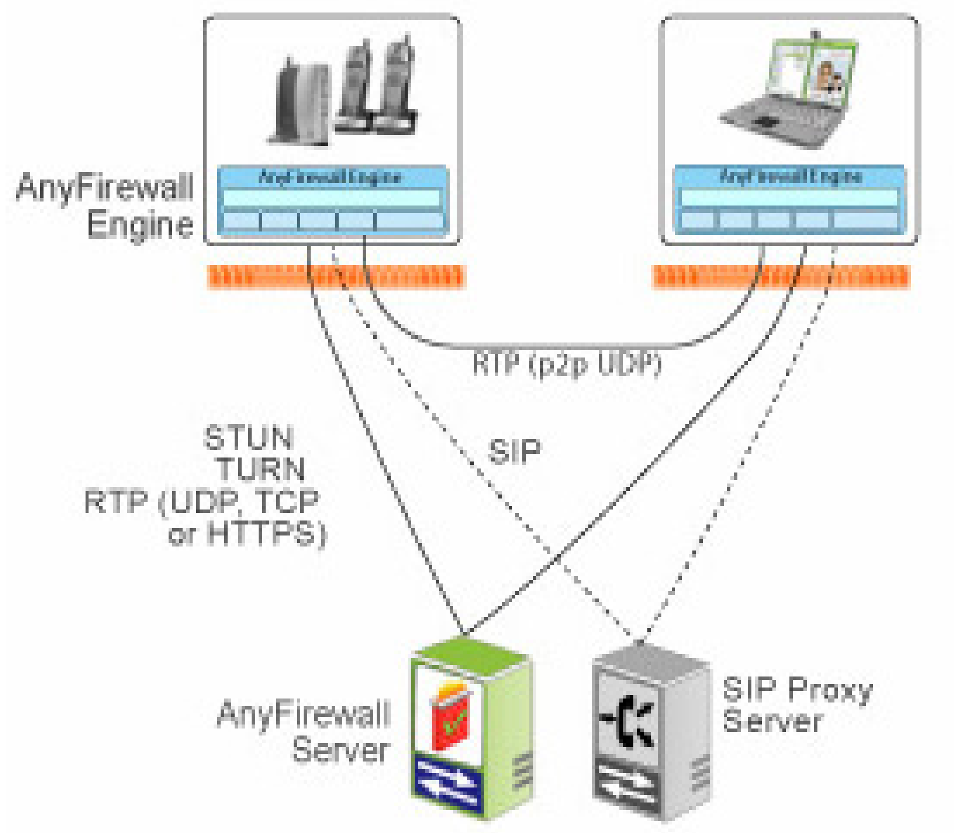
Reliability issues

- Wide device mobility using link layer technologies (WiFi, WiMax, 3+G)
- Limited network and device mobility using network layer protocols (MobileIP)
- Limited transport switching based on ad hoc solutions (Rocks, FT-TCP, TCP-Migrate)



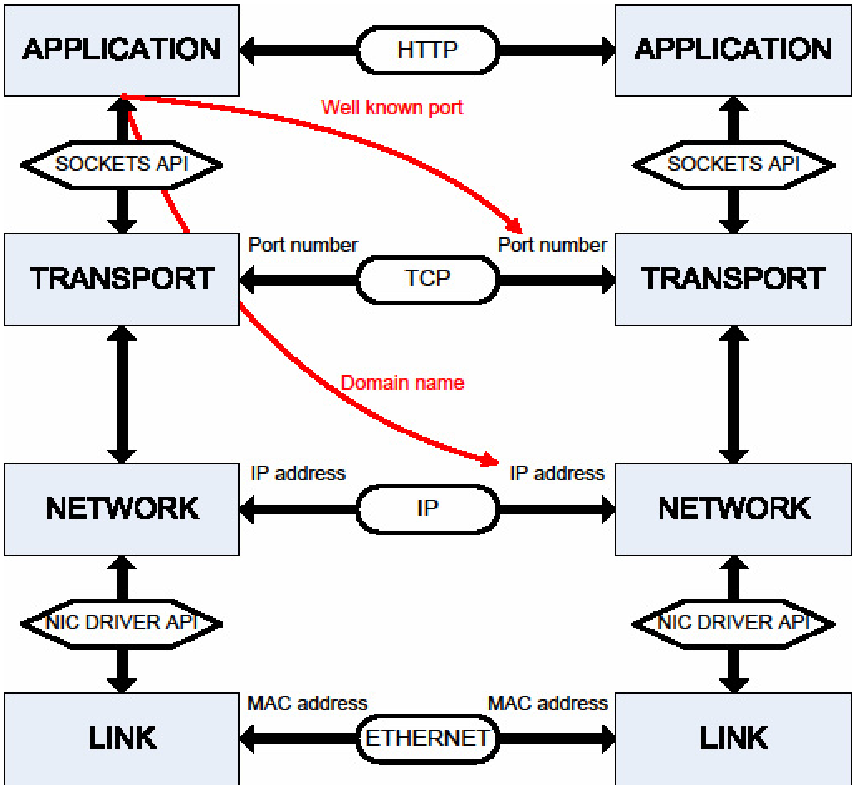
Complexity issues

- IPv4/v6 gateways
- Proxies
  - Web, SIP, e-mail
- Firewalls
- NATs
  - UPnP (IGDP), ICE, STUN, TURN



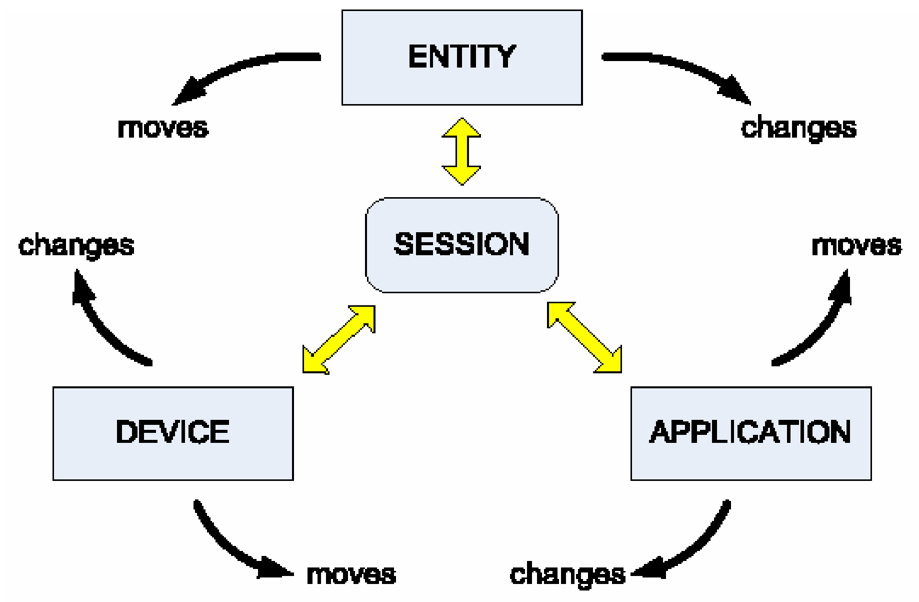
Frozen stack

- Application bound to a locator (IP@)
- Application bound to a transport protocol (protocol n°)
- Application bound to a multiplexing ID (port n°)
- Connection is broken if anything changes



CLOAK paradigm

- A *communication* can switch devices, entities and applications at will and on the fly (when it makes sense)
- A *session* manages all the control information needed for the communication

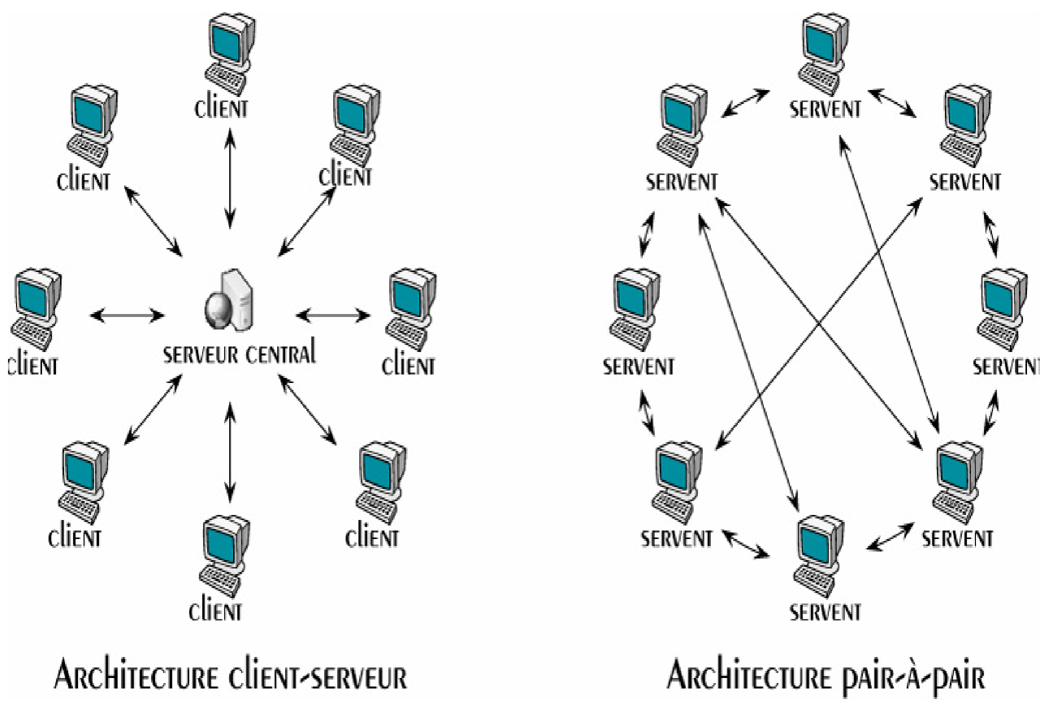


CLOAK concepts

- P2P links
  - Dumb network
- Overlays
  - E2E model
  - Unique addressing space
- Names
  - Definable name spaces
- DHTs
  - Autonomous storage

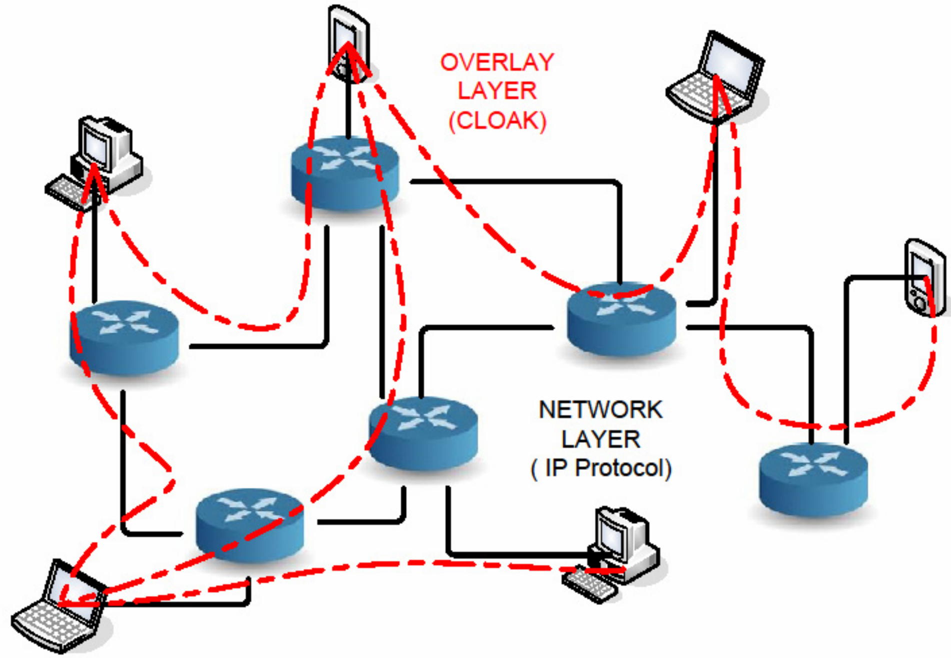
Peer-to-Peer networking

- Device can work as a client and as a server
- Any device can interact with any other
- No role hierarchy
- No constrained topology
- No single point of failure
- No device or link overload



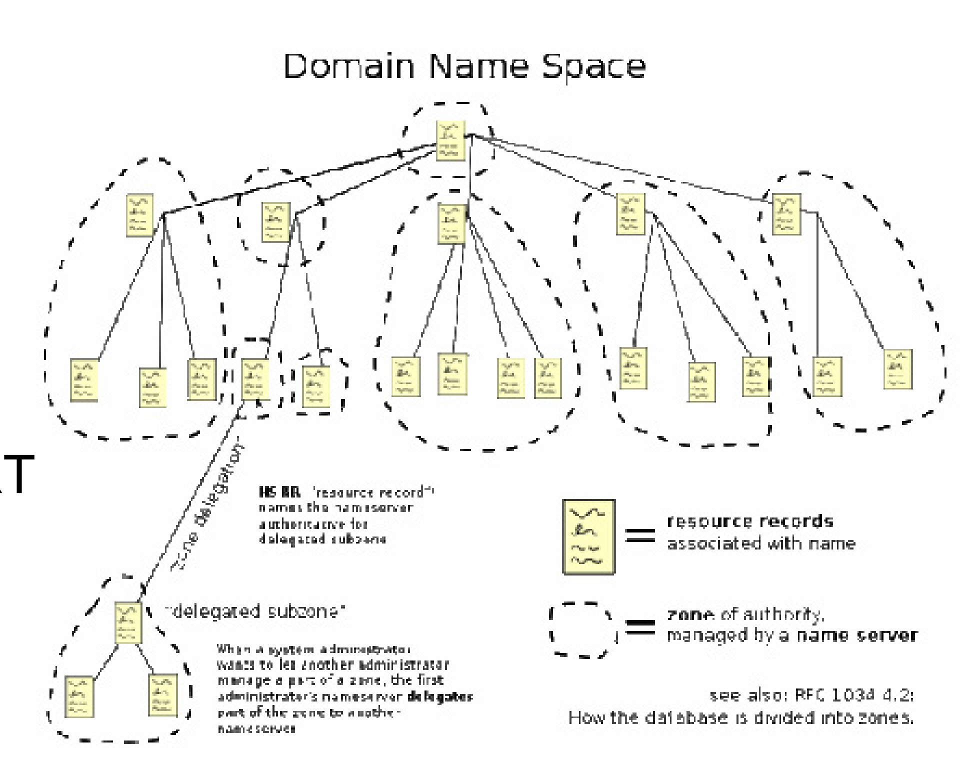
Overlay networking

- Terminal devices connect to some others creating virtual links
- Devices with 2+ links play the role of routers
- Devices form a new virtual network called an *overlay*



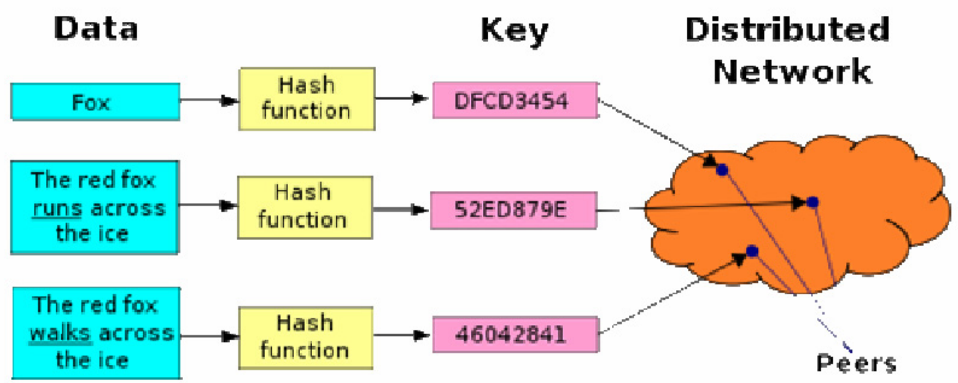
Names

- The Internet gives IDs
  - Host names (server.abc.gov)
  - Service names (smtp)
  - E-Mail and VoIP @s (bob@abc.edu)
  - URLs (http://www.abc.com/path/file.html)
  - URIs (urn:isbn:0-486-27557-4)
- All IDs contain IP@s and PORT n° in disguise!
- The DNS is mostly static
- ID / Locator separation is needed: names
- Dynamic binding is needed



Distributed Hash Tables

- A Hash Table is a directory/map storing {key,value} pairs
- HT split, replicated and located on many devices = DHT
- Queries in log(n)
- Greedy key-based routing
- Consistent hashing

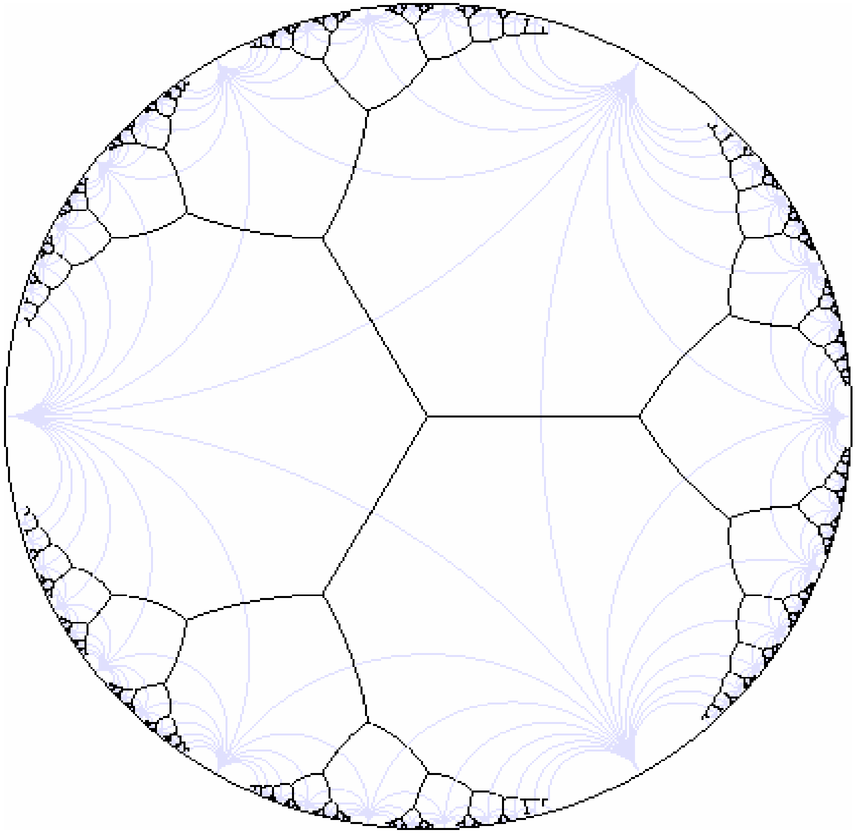


CLOAK design

- Addressing (distributed in each peer)
- Routing (local knowledge / compact)
- Naming (for devices and entities)
- Binding (DHT)
  - {device name,device address}
  - {entity name,device name}
- Steering (dynamic binding)

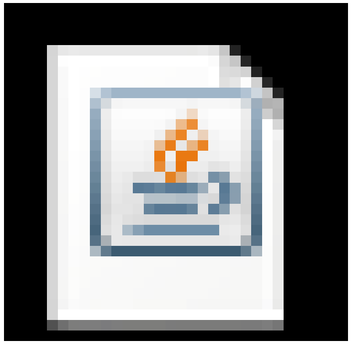
The Poincaré disk

- A model of the hyperbolic plane
- The open unit disk represents the plane
- The unit circle represents points at infinity (horizon)
- Lines are arcs cutting the horizon at right angle



Addressing in the Poincaré disk

- A model of the hyperbolic plane
- The open unit disk represents the plane
- The unit circle represents points at infinity (horizon)
- Lines are arcs cutting the horizon at right angle
- Applet (thanks to Don Hatch!)



hyperb.jar

Routing in the Poincaré disk

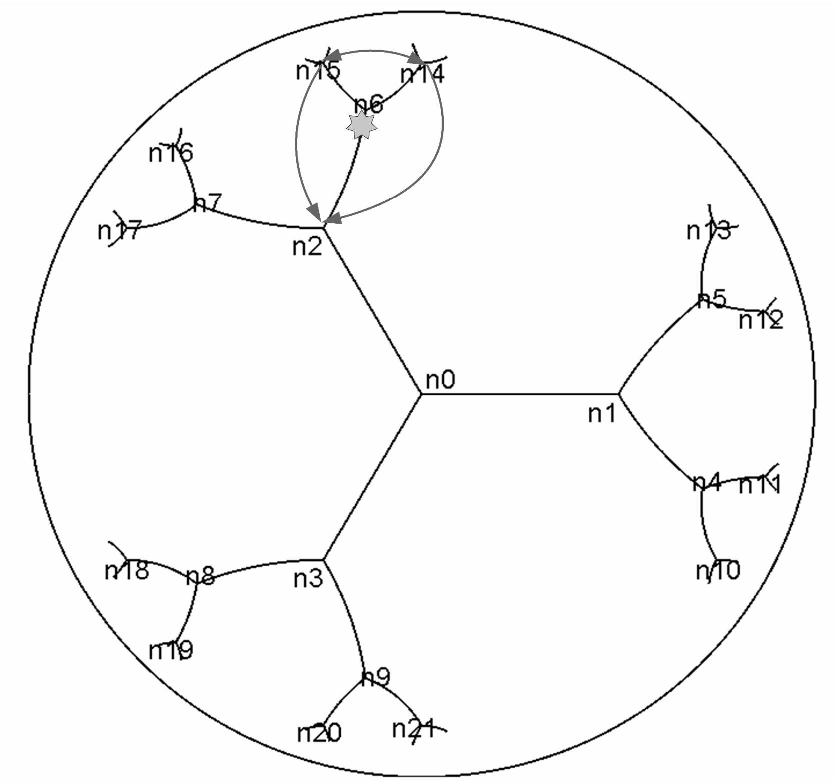
- Greedy routing by using hyperbolic coordinates in the unit disk (represented by a complex number)
- Hyperbolic distance *D(a,b)* between *a* and *b* is given by:

$$D(a,b) = \operatorname{arg\,cosh}\left(1 + \frac{2|a-b|^2}{\left(1-|a|^2\right)\left(1-|b|^2\right)}\right)$$

- When a packet to d enters peer p:
  - Compute *D(n,d)* for each neighbor peer *n*
  - Choose the next hop *n* that has the lowest *D*

Alternate routing

- On peer or link failure
  - Flush @s of descendants (root can't die)
  - Replace the missing and give same @ (must implement @->name)
- Maintain connectivity
  - Connect to ascendents
  - Connect to siblings

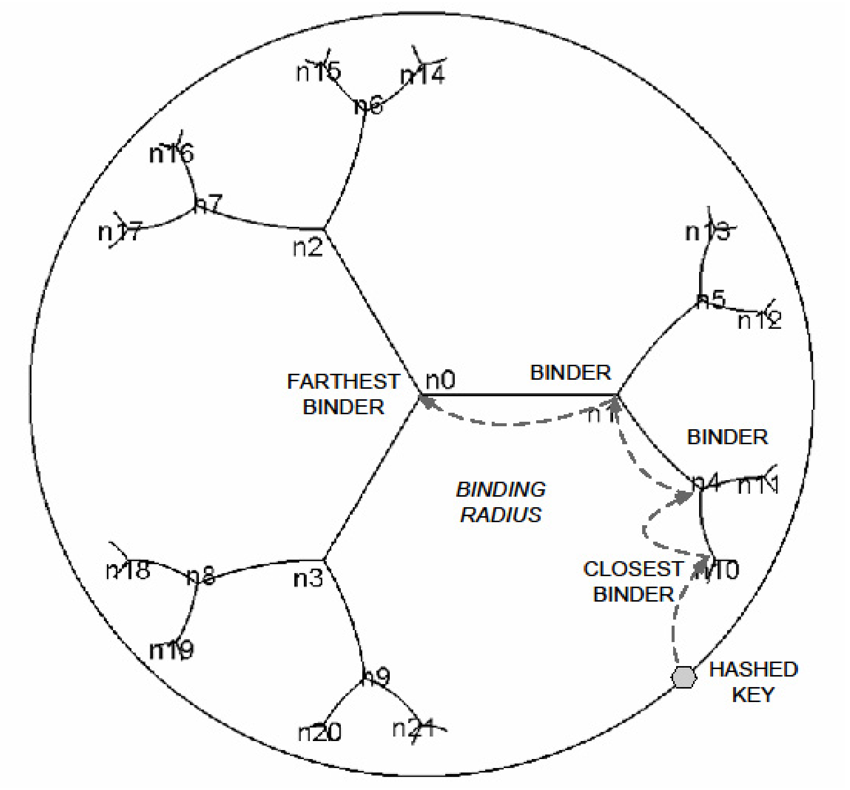


Naming

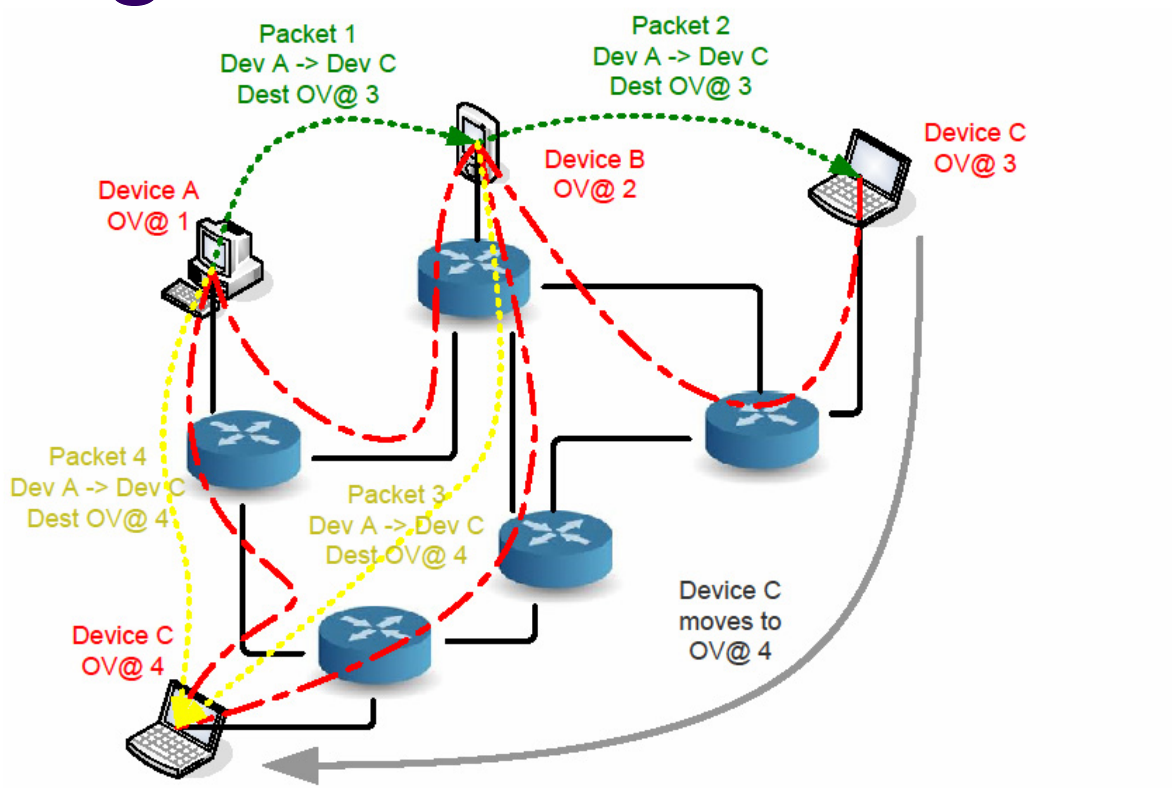
- Names must be defined by the users of the overlay for entities and devices
  - They must be unique inside the overlay
- Names are stored in the DHT formed by the peers of the overlay
- Group names can be defined
  - For multicast
- Entities names can be layered (aliases)

Storing and solving names

- A name is hashed as a key with SHA-1
- The key is normalized as *k* between [0,1]
- The normalized key *k* is mapped to a virtual point on the circle
  - $x = \cos(2\pi k)$
  - $y = \sin(2\pi k)$
- The (name,@) is stored in the peer closest to the virtual point



Steering

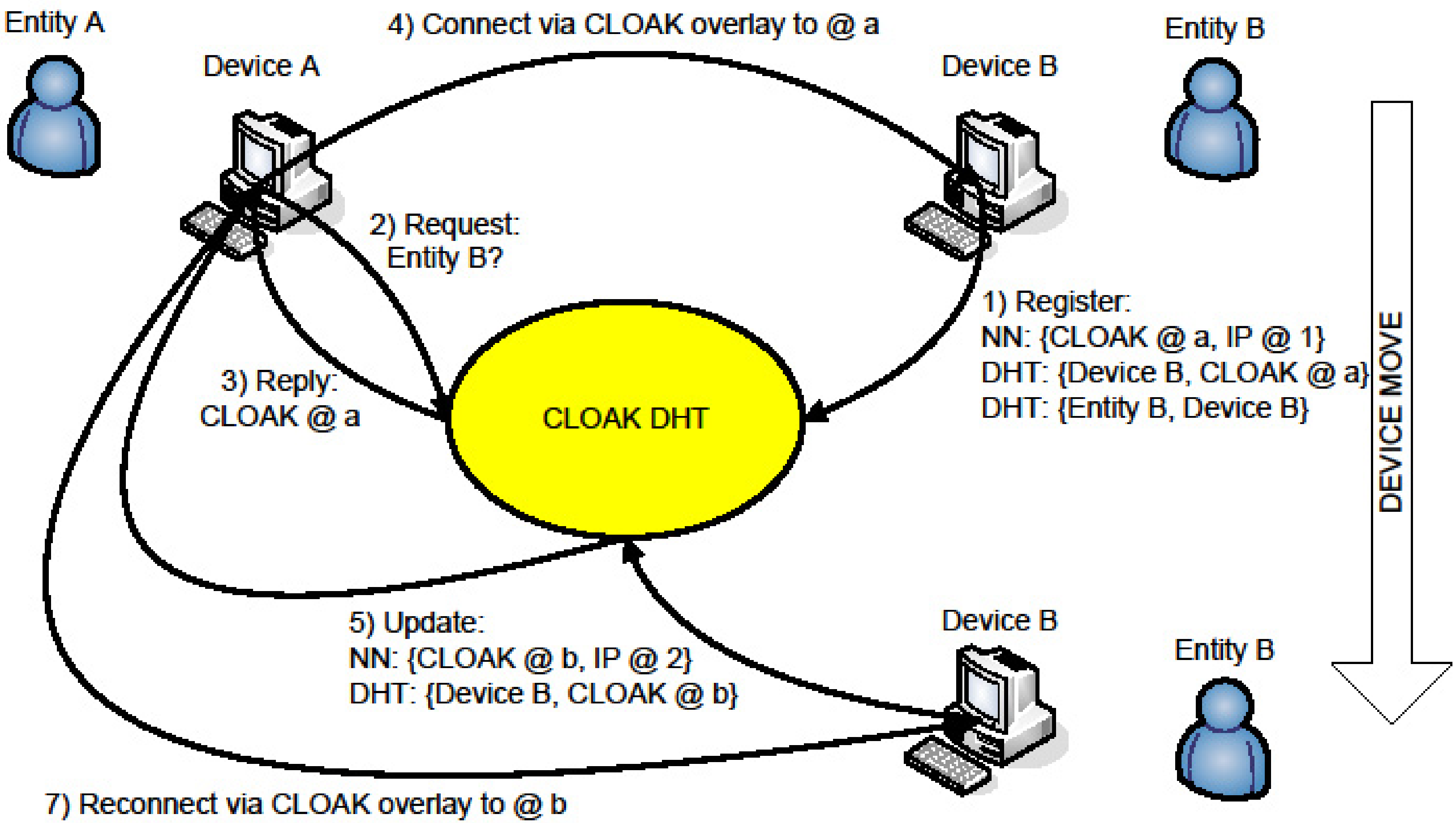


- If an overlay @ becomes invalid, reply packets contain new @ and intermediate peers can reroute on the fly until source peer can reroute
- This mecanism also enables multicast capability

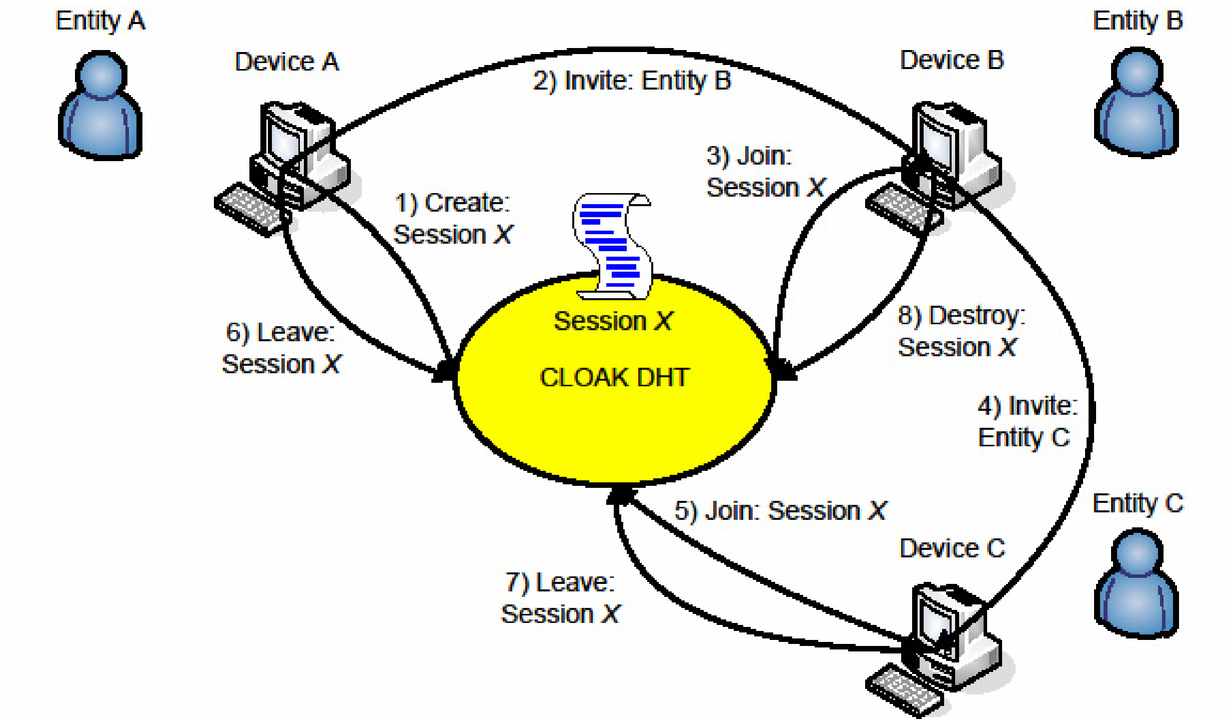
Communications in the overlay

- Bootstrap into the overlay by setting transport layer connections to peers (neighbor peers)
- Obtain an overlay address from a neighbor peer
- Identify oneself in the overlay with unique device and entity names
- Create a session
- Invite 1+ peer(s) to communicate with
- Set overlay layer connections
- Send data streams to the session members

Identification and localization

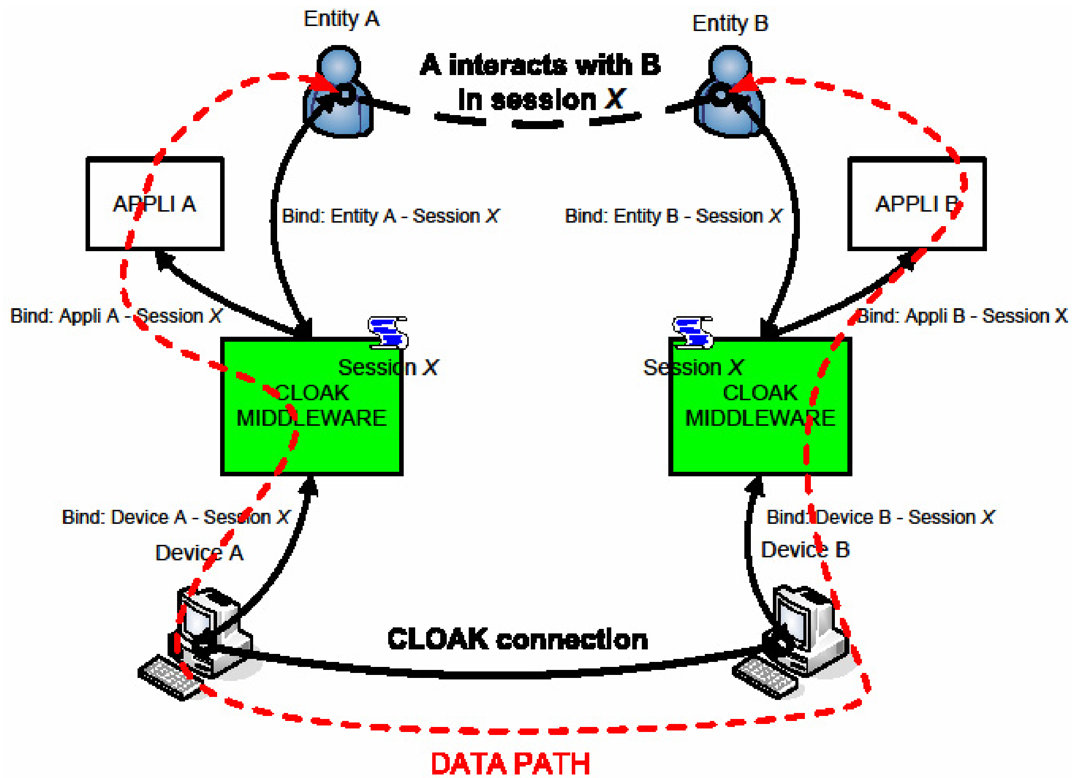


Session



- A session manages a communication between 2+ overlay peers
- Ends when all peers have quit

Interaction

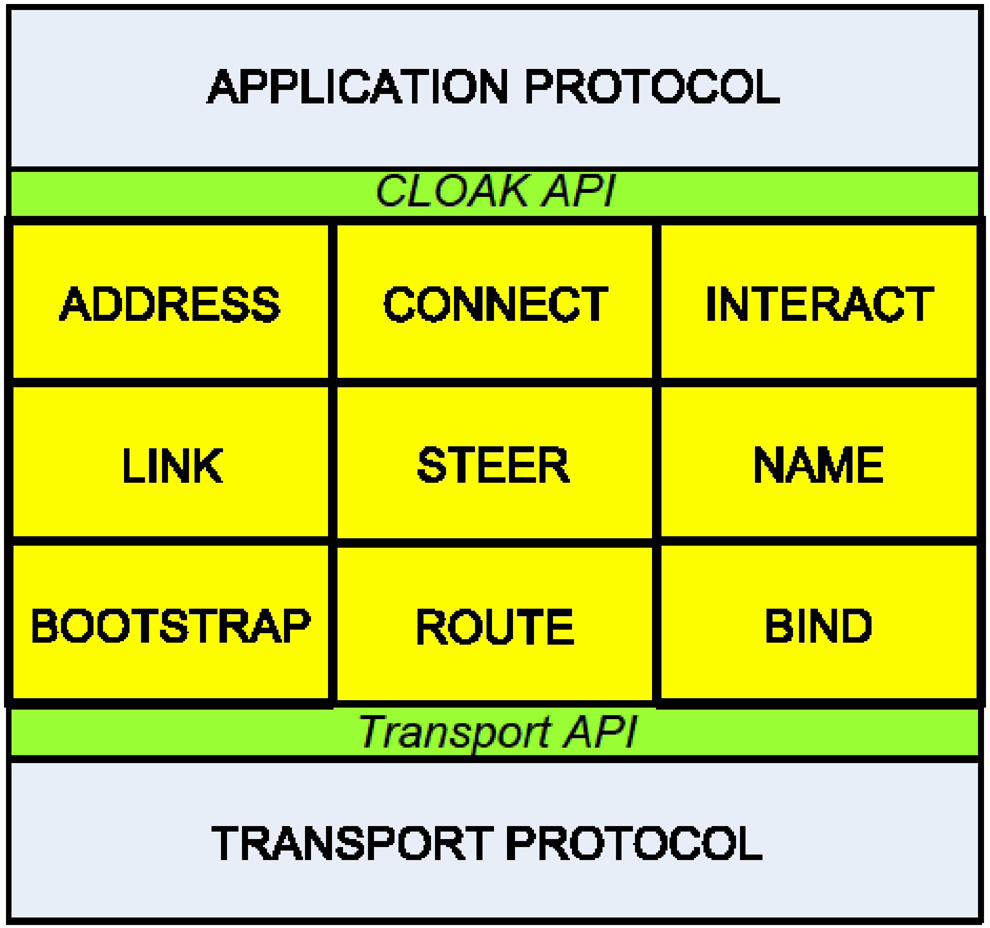


- A connection uses a quadruplet {stream,application,entity,device} at each endpoint
- All connections are stored in the session

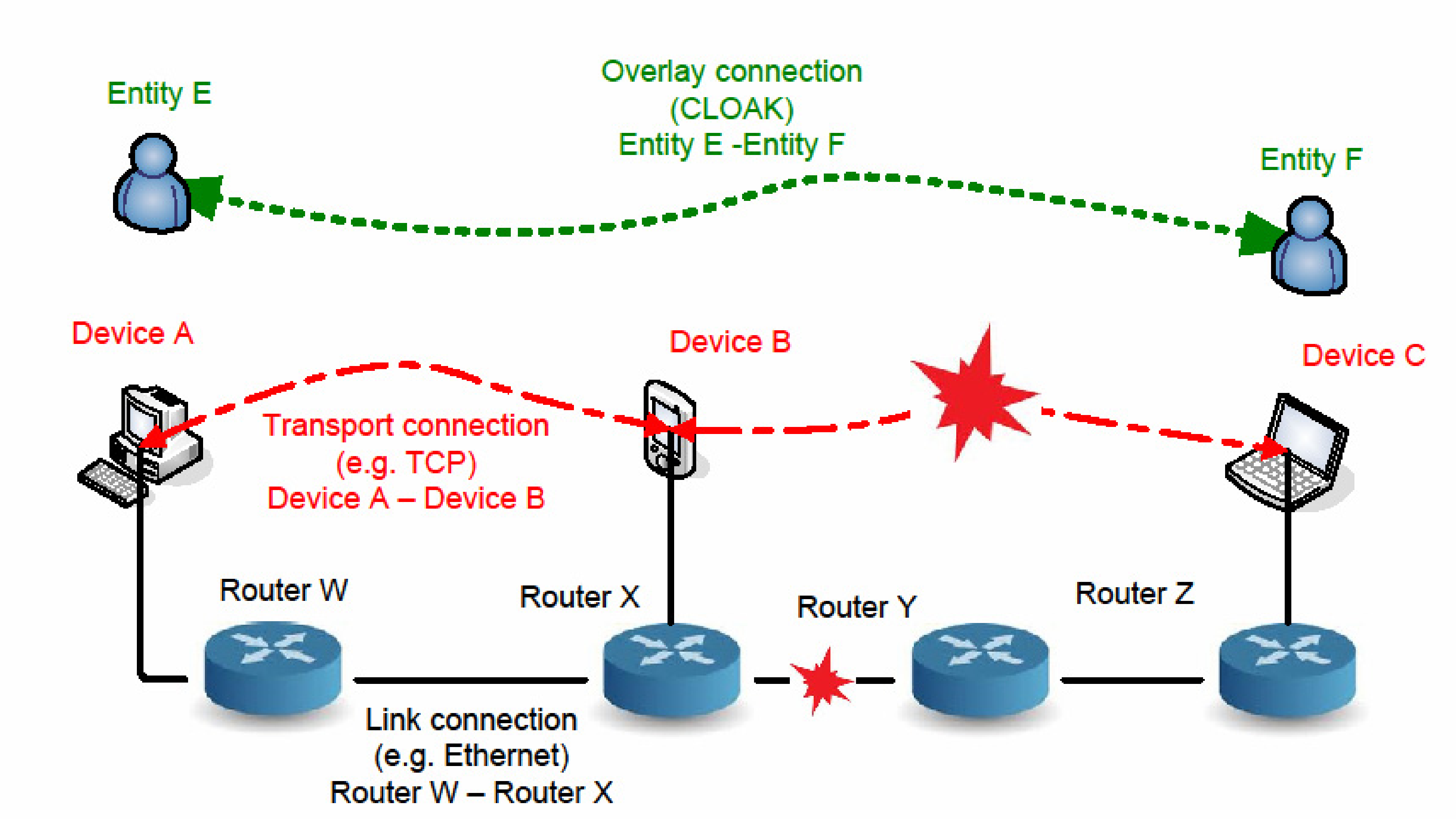
CLOAK architecture

- Modules of the middleware
- Layers of connections
- Protocol stack
- Protocol headers and encapsulation

Modules

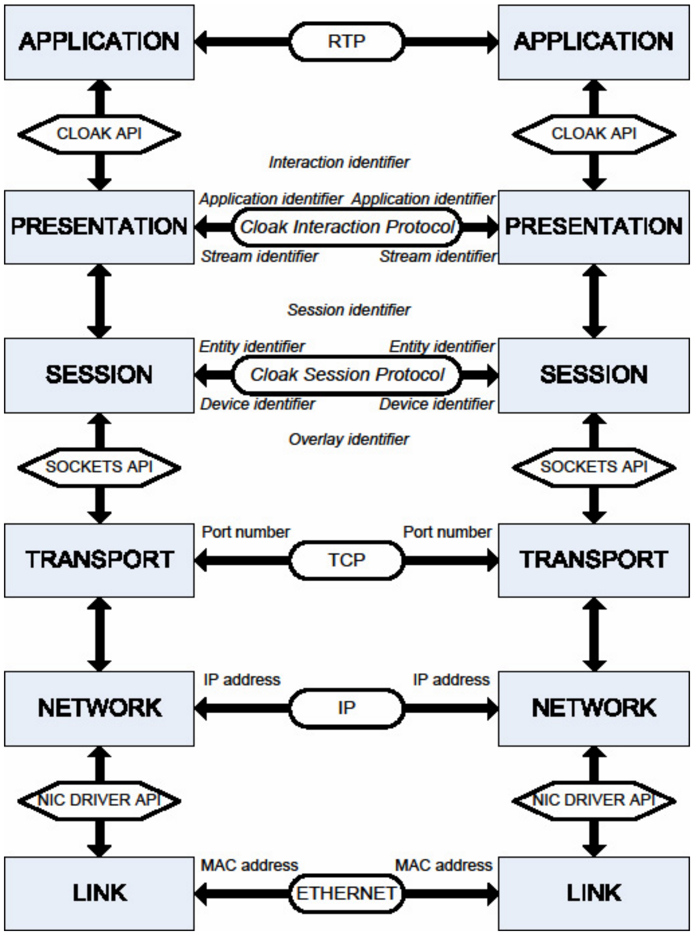

APPLICATION PROTOCOL
CLOAK API
ADDRESS
CONNECT
INTERACT
LINK
STEER
NAME
BOOTSTRAP
ROUTE
BIND
Transport API
TRANSPORT PROTOCOL

Connections

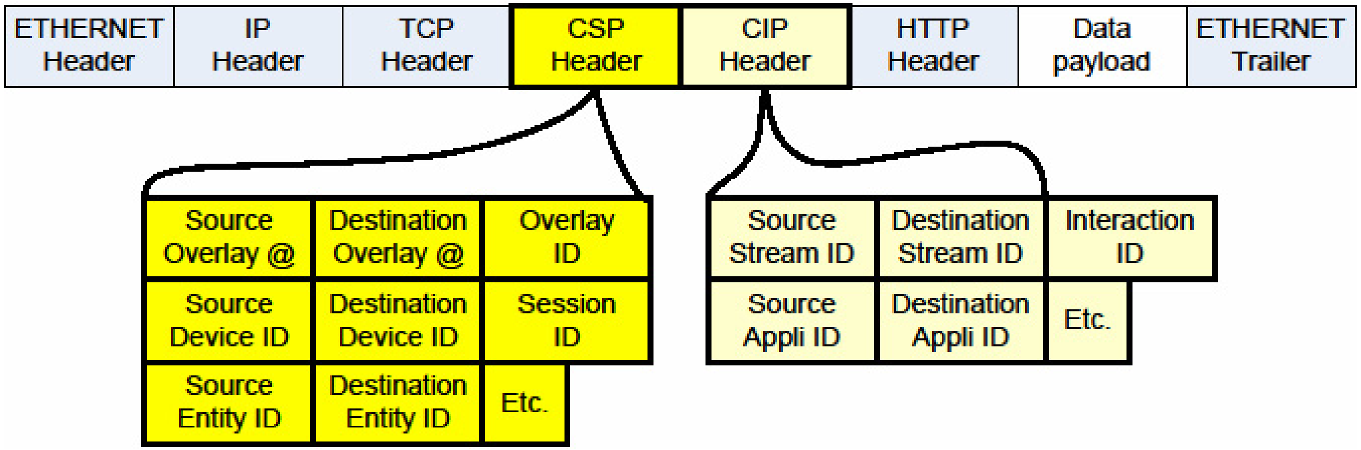


Stack

- Adds new protocols between transport and appli layer protocols
- Adds new identifiers and names
- Adds a new API
- Application is bound by virtual IDs but network IDs (IP@, protocol n°, port n°) can change



Encapsulation



- 2 additional headers and protocols
- Overlay @ for routing in the overlay and moving devices
- Device ID for switching devices and moving entities
- Entity ID for switching entities
- Stream ID for virtual port numbering on the entity
- Application ID for selecting or switching applications

Usages

- Mobile and switchable applications
- Scalable and reliable dynamic VPNs
- Convergence layer for IPv4, NATs and IPv6
- Anonymizing layer for darknets
- Adaptive transport protocol switching and chaining
- Definition and use of new namespaces

Cloakable applications

- Messaging
  - Contact anywhere
  - E2E confidentiality
- Conferencing
  - Talk on the move
  - Multicast capability
- Sharing
  - Network anonymity
  - P2P confidentiality

Cloakable applications cont'd

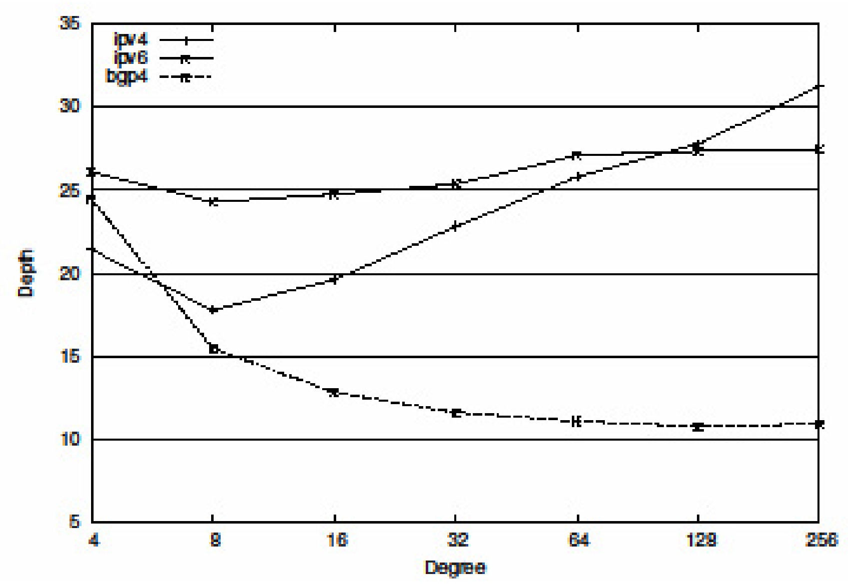
- Streaming
  - Watch on the move
  - Redirect stream on the fly to another device or user
- Socializing
  - Secure F2F
  - Contact anywhere

Simulations

- Static simulation where each peer session length = simulation duration
- On Internet maps (IPv4, IPv6 and BGP) where each node is a peer
- Variation of the degree of the addressing tree from 4 to 256
- One packet sent from every peer to every others (all paths evaluated)

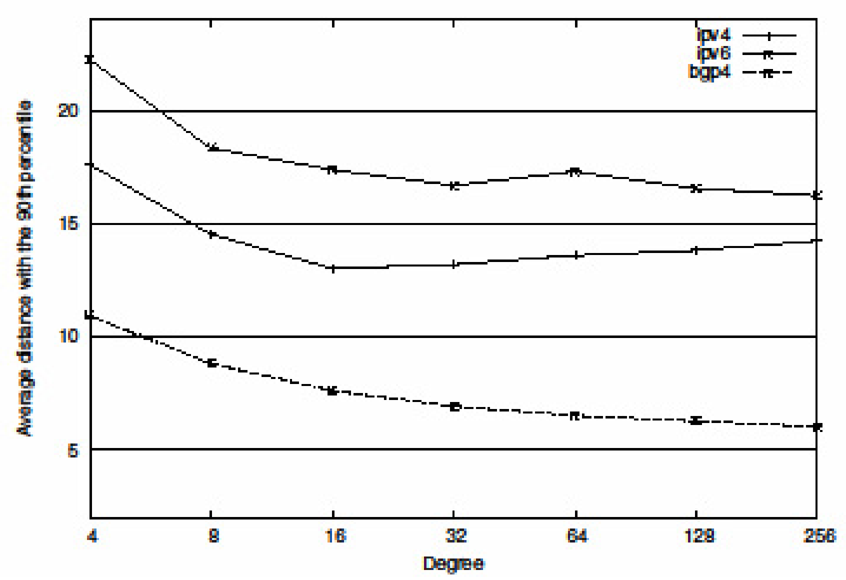
Depth

- Depth of the addressing tree for covering the maps
- Strongly depends on the value of the degree
- Strongly depends on the map type



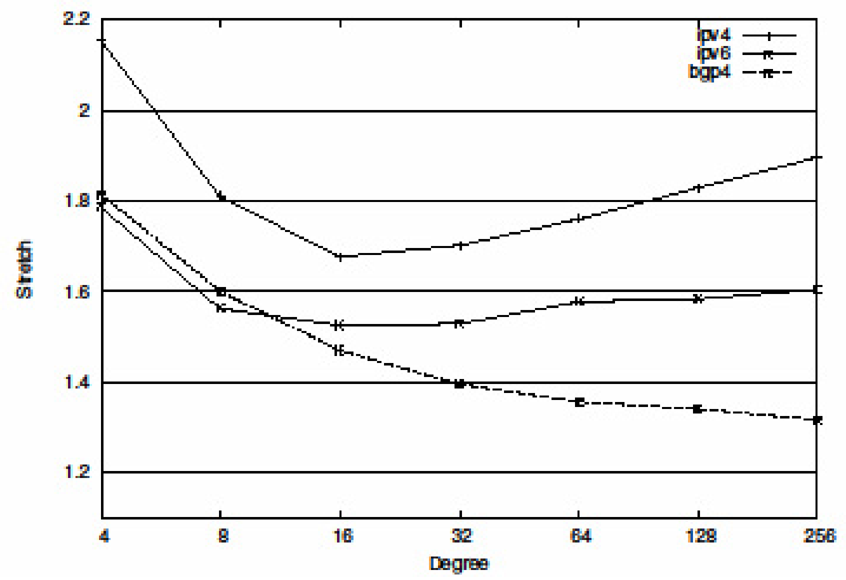
Path length

- Average path length between all nodes
- Proportional to the size of the map
- Correlated to the degree
- Diminishing return for IP maps with optimal around 16



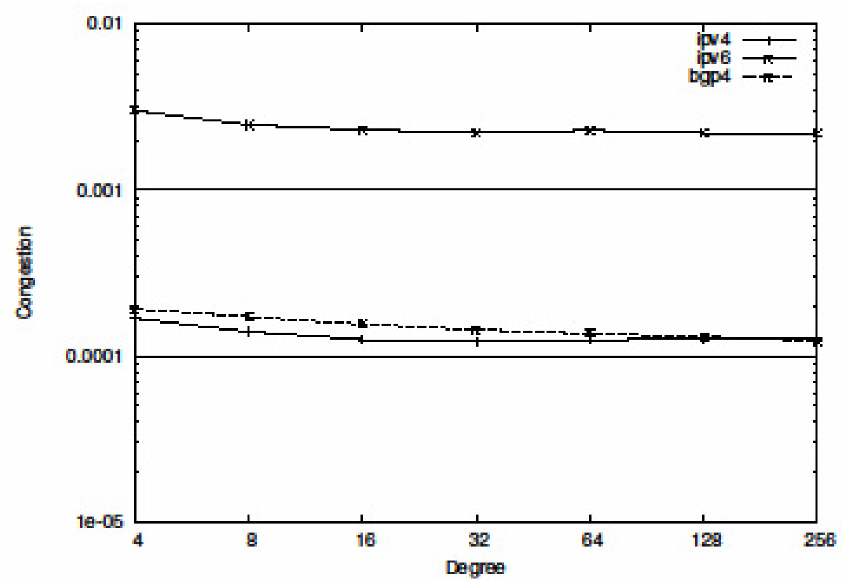
Stretch

- Remains below 2.2
- Correlated to the degree
- Strong diminishing return for IP maps with optimal around 16



Congestion

- Very low on average
- Proportional to the size of the map
- Uncorrelated to the degree



Conclusion

- Design of an overlay architecture that provides new features to applications
  - Mobility, flexibility, security, autonomy
- Middleware that has nice properties
  - Scalable, reliable, simple
- Implementation in C++ in *nem*
- Simulation results on 10k+ sized Internet maps show that it works with acceptable performances

Future work

- Define APIs
- Code implementations in C++ and Java
- Deploy in user space libraries
- Port in test application (text chat)
- Evaluate in virtualized environments
- Integrate in kernel space modules
- Port in real applications (IM, TV)
- Deploy in the Internet

The end

- Thank you!
- Any questions?