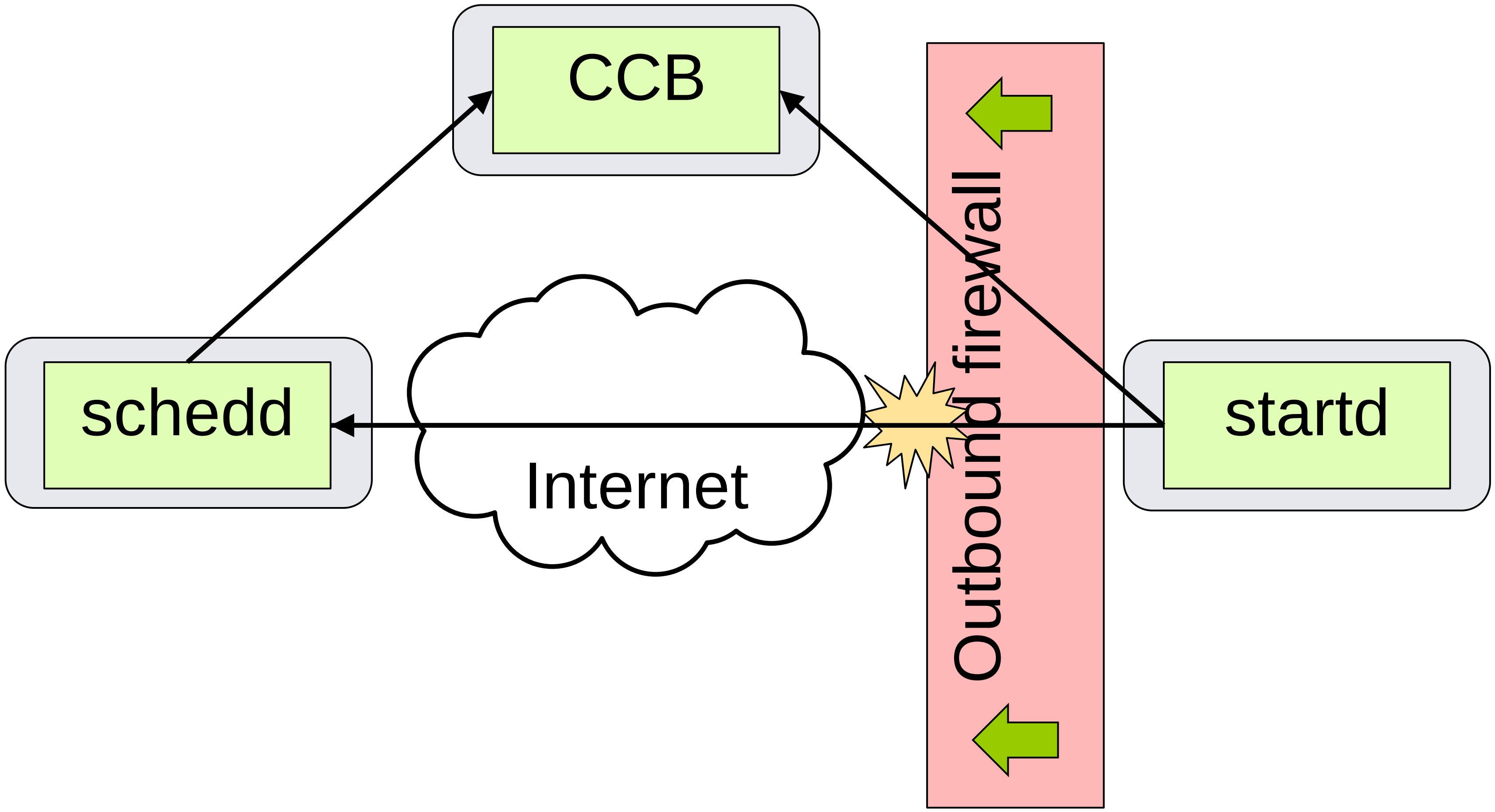
CENTER FOR
HIGH THROUGHPUT
COMPUTING

The Future of HTCondor's Networking
or:
How I Learned to Stop Worrying and Love IPv6

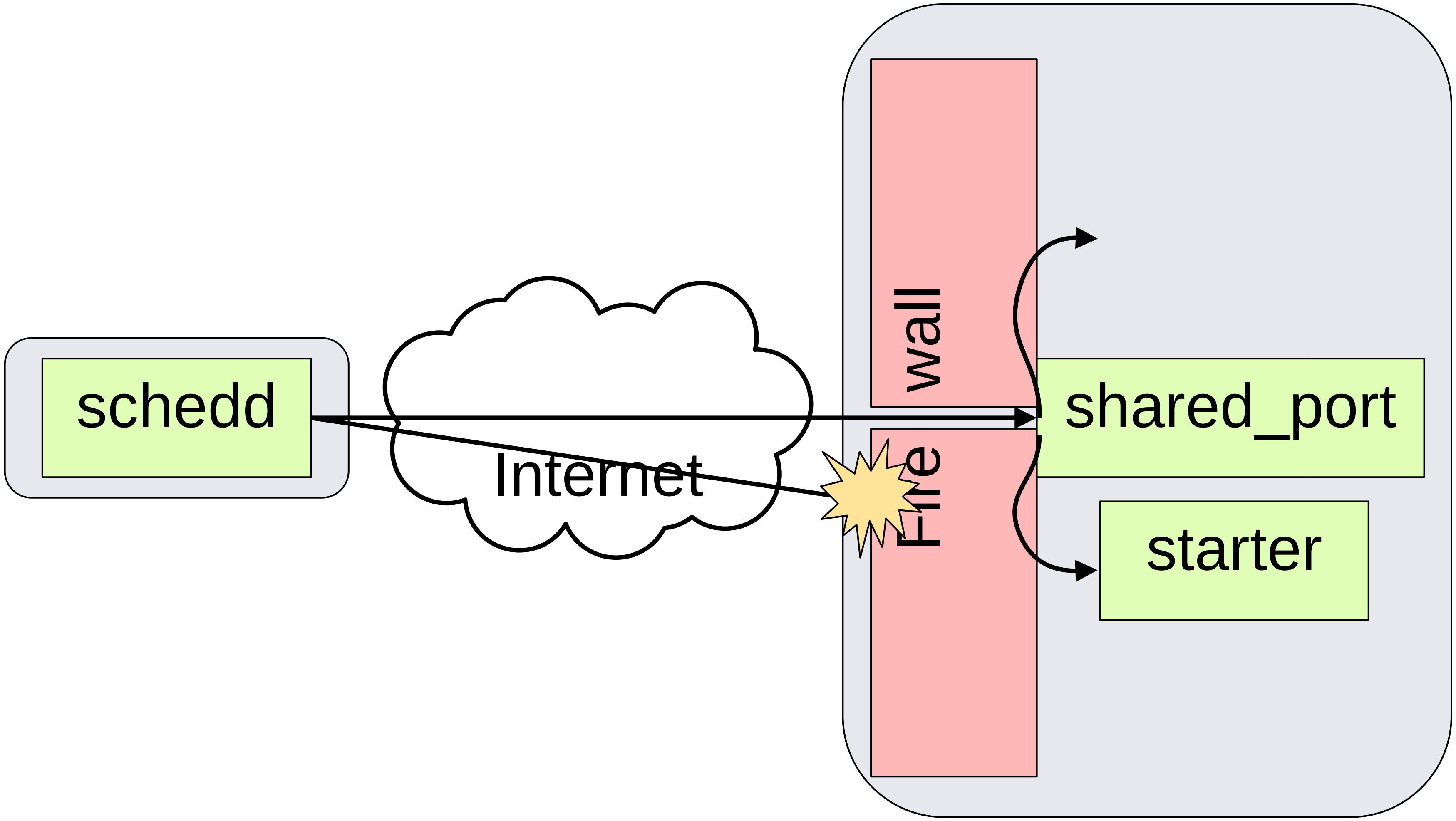
Alan De Smet
adesmet@cs.wisc.edu

8.0: The Past

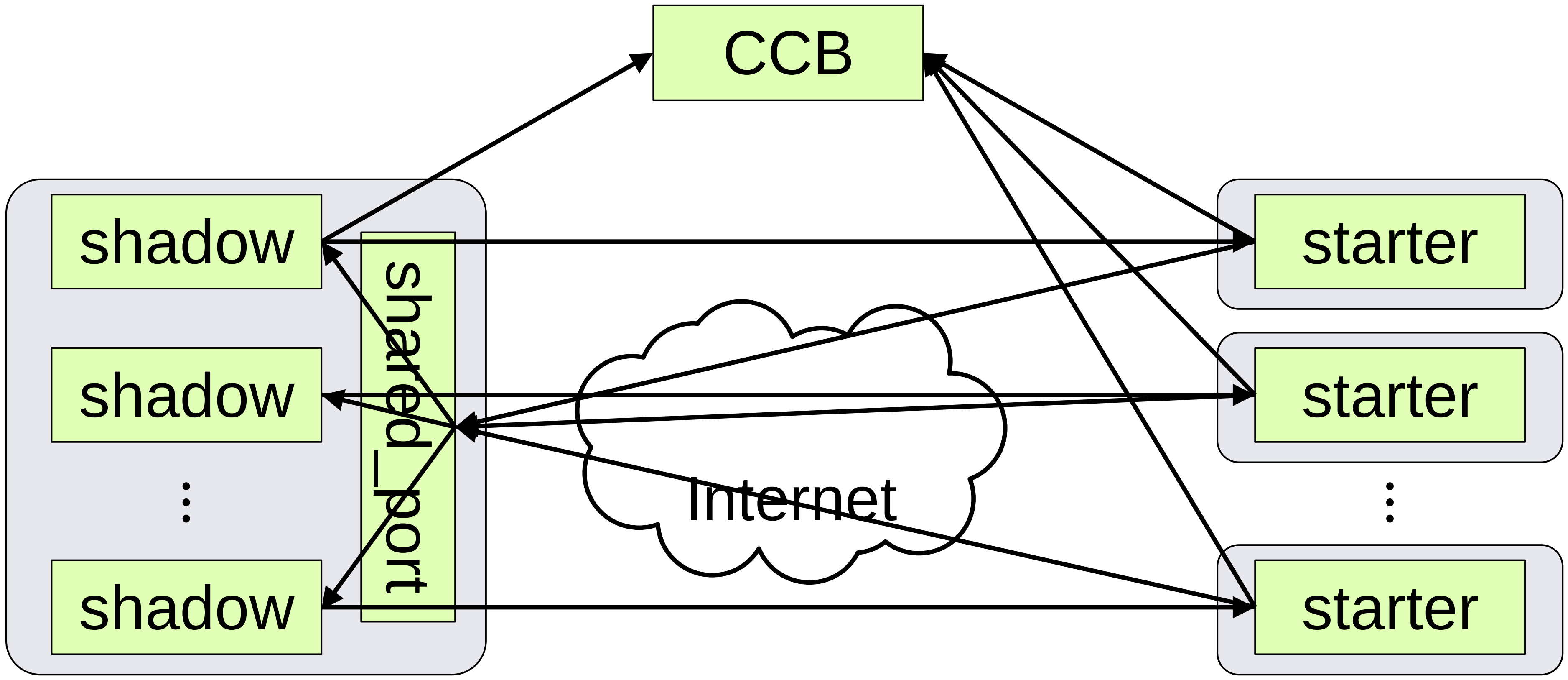
CCB: Condor Connection Broker



condor_shared_port



Beating the 65,535 limit



Do-It-Yourself Name Service

- Minimal DNS dependency
  - Working to reduce!
- Centralized service: condor_collector
- More than addresses; routes!
  - CCB, shared_port

IPv6 support

- IPv4 by default
- `ENABLE_IPV6=TRUE`
  - Disables IPv4
  - Lots of odd limitations
- Only one interface (sorta)

8.1 & 8.2: The Present

- IPv4 by default
- **`ENABLE_IPV6=TRUE`**
  - ***IPv4 stays on!***
  - Lots of odd limitations
- Using IPv6 today? Also do
  - **`ENABLE_IPV4=FALSE`**

8.1 & 8.2: The Present

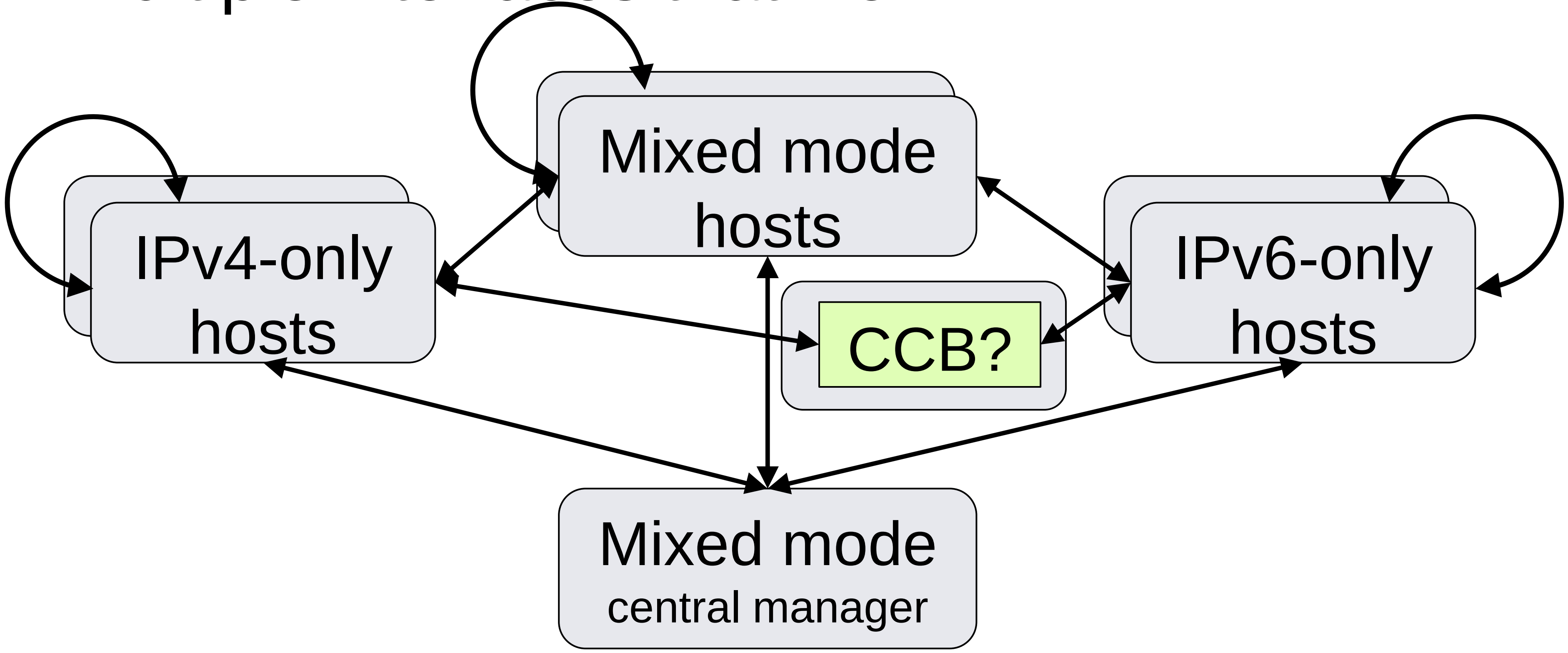
- Mixed mode IPv4 and IPv6!
- Sorta
- Kinda
- Not really

8.1 & 8.2: The Present

- Mixed mode IPv4 and IPv6?
- Simple queries
- You *might* get matchmaking working
  - No support
- Multiple interfaces are supported, but…
- *…only the IPv4 address is advertised*

8.3 & 8.4: The Future

- Mixed mode IPv4 and IPv6! Really!
- Mixed pools! (Mixed matchmaking?)
- Multiple interfaces that work!



8.3 & 8.4: The Future

- Mixed mode IPv4 and IPv6! Really!
- Mixed pools! (Mixed matchmaking?)
- Multiple interfaces that work!
- Complete overhaul of address representation!

A complete what now?

- Complete overhaul of address representation!

Addresses Today

- Today: Sinful Strings. Simple, elegant

```
<173.194.46.96:80>
<192.168.1.55:9618?PrivNet=example.com
&PrivAddr=192.168.1.55&sock=1567_808b_
3&CCBID=173.194.46.96:80#381>
```

Addresses Today

- New features bolted on
  - Backward compatible
  - Minimal changes / Maximal stability
  - Fast development

```
<192.168.1.55:9618?PrivNet=example.com
&PrivAddr=192.168.1.55&sock=1567_808b_
3&CCBID=173.194.46.96:80#381>
```

- IPv6 would double the complexity
- What about IPv8, or CCB2, or Unix Domain Sockets, or…

Future Addresses

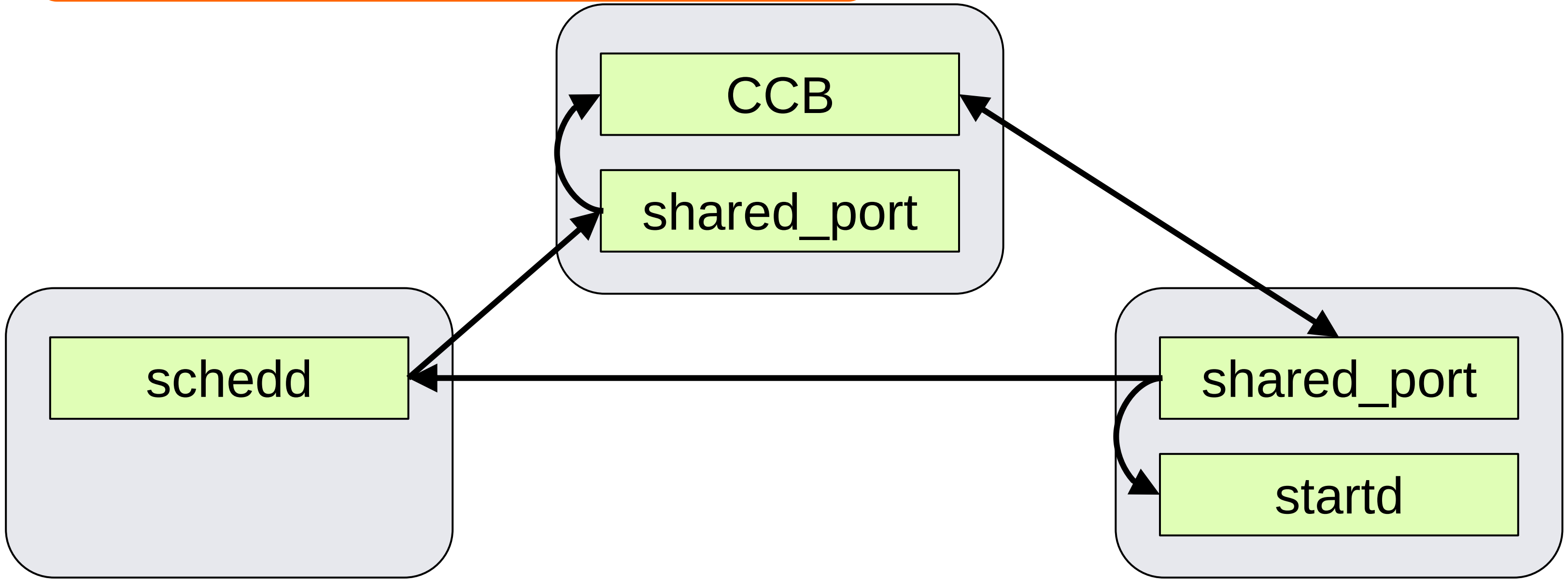
- Making a new way to describe addresses
- Extensible, support complex data structures
- Generalize specific techniques from Sinful strings

- Backward compatible for one major version: support both old and new versions

Generalizing Lessons

Multi-Stage Routing

```
<192.168.1.55:9618?
CCBID=173.194.46.96:80#381%3F
     sock%3D917_aa8b_3&
sock=1567_808b_3>
```



Multi-Stage Routing

- Proxies
  - CCB, shared_port (sorta)
  - CCB2?, SOCKS?, HTTP CONNECT?
- Special setup
  - VPNs? SDN dynamic routes? Port knocking?
- Multiple routes simultaneously?
  - budget channel bonding

Multi-Stage Routing

(IPv4, 42.82.111.3, 9618), (SharedPort, 381_382b_3), (CCB, 37349), (SharedPort, 5491_b8c1_1)

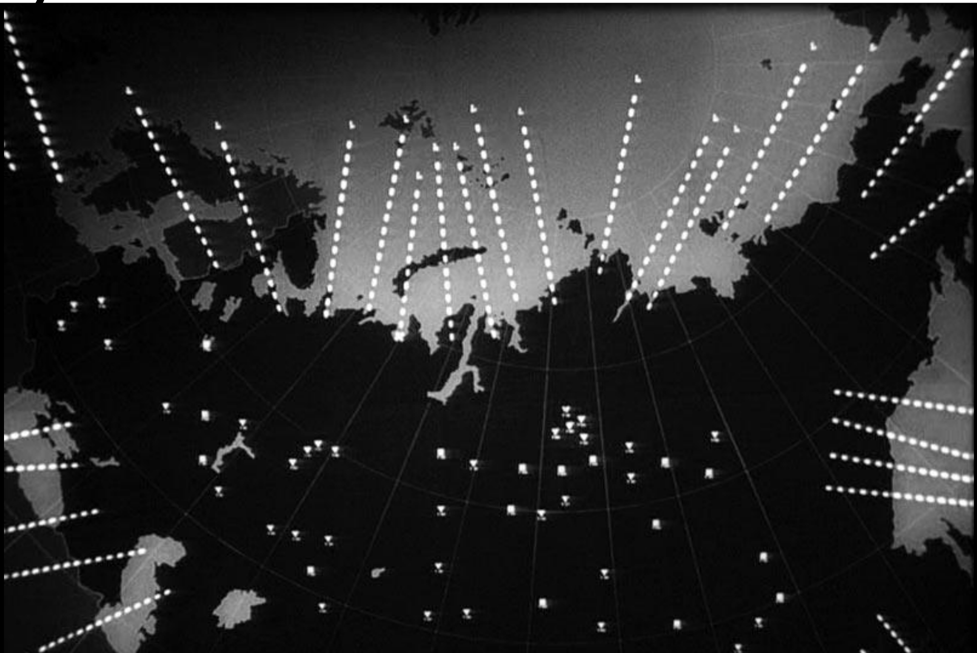
1. IPv4: Connect to 42.82.111.3 port 9618
2. shared_port: ask for 381_382b_3
3. CCB: ask for 37349, wait for reverse connection
4. shared_port: ask for 5491_b8c1_1
5. startd!

› Might be a DAG…

Multiple Routes

```
<192.168.1.55:9618?
PrivNet=example.com&
PrivAddr=192.168.1.55&
CCBID=173.194.46.96:80#381>
```

› We may not be able to use them all



Multiple Routes

- Different protocols:
  - IPv4, IPv6
  - IPv8?, Unix Domain Sockets?
- Accessibility:
  - public, private
  - multiple private networks? fast versus cheap?

Multiple Routes

- A list of routes, identified by starting network

example.com, (IPv4, 192.168.1.55, 9618), (SharedPort, 5491_b8c1_1)

public, (IPv4, 173.194.46.96, 80), (CCB, 381), (SharedPort, 5491_b8c1_1)

Bi-directional Routing

- Matchmaking
  - Can the schedd reach the startd?
  - What if the startd needs to reverse the connection?
- Today: no support
- Advertise which networks and protocols the daemon can connect to
  - Private:UW, IPv4
  - Internet, IPv6
  - Internet, IPv4

Client Route Selection

- A client has a choice of routes, which one?
- RFC 6724: Default Address Selection for Internet Protocol Version 6 (IPv6)?
  - We're not using DNS
  - We're *way* weirder than DNS

CENTER FOR HIGH THROUGHPUT COMPUTING

Client Route Selection

- Client obtains (from the collector) the list of the endpoint's inbound paths.
- Client discards inbound paths which it can not reach.
  - Client can reach a path if one of its outbound paths terminates with a protocol and a network name identical to the protocol and network name of the inbound path in question.
- Client discards any remaining reversing paths that can not reach it.
  - A reversing path can reach the client if the reachable set of each reversing step includes a protocol and network name pair identical to one terminating one of the client's return paths.
- Client constructs the complete path corresponding to each remaining inbound path.
- The client tries one or more of the complete paths, in whatever order or concurrency it finds appropriate. For each path it tries, the client interacts with each step appropriately. For a reversing step, this will include supplying a list of reachable return paths, which the client may need to configure.

We're working on it

Toward a bold new future of glasnost between IPv4 and IPv6!