NAT-XC
draft-moore-nat-xc

Keith Moore
moore@network-heretics.com

IETF 74 - BEHAVE WG meeting
25 March 2009

Overview

- Motivations for NAT-XC
- Brief description and architecture
- What the protocol looks like
- Protocol usage
- Deployment scenarios
- NAT-XC as unifying architecture
- Conclusion

Motivations

- Ease transition to IPv6

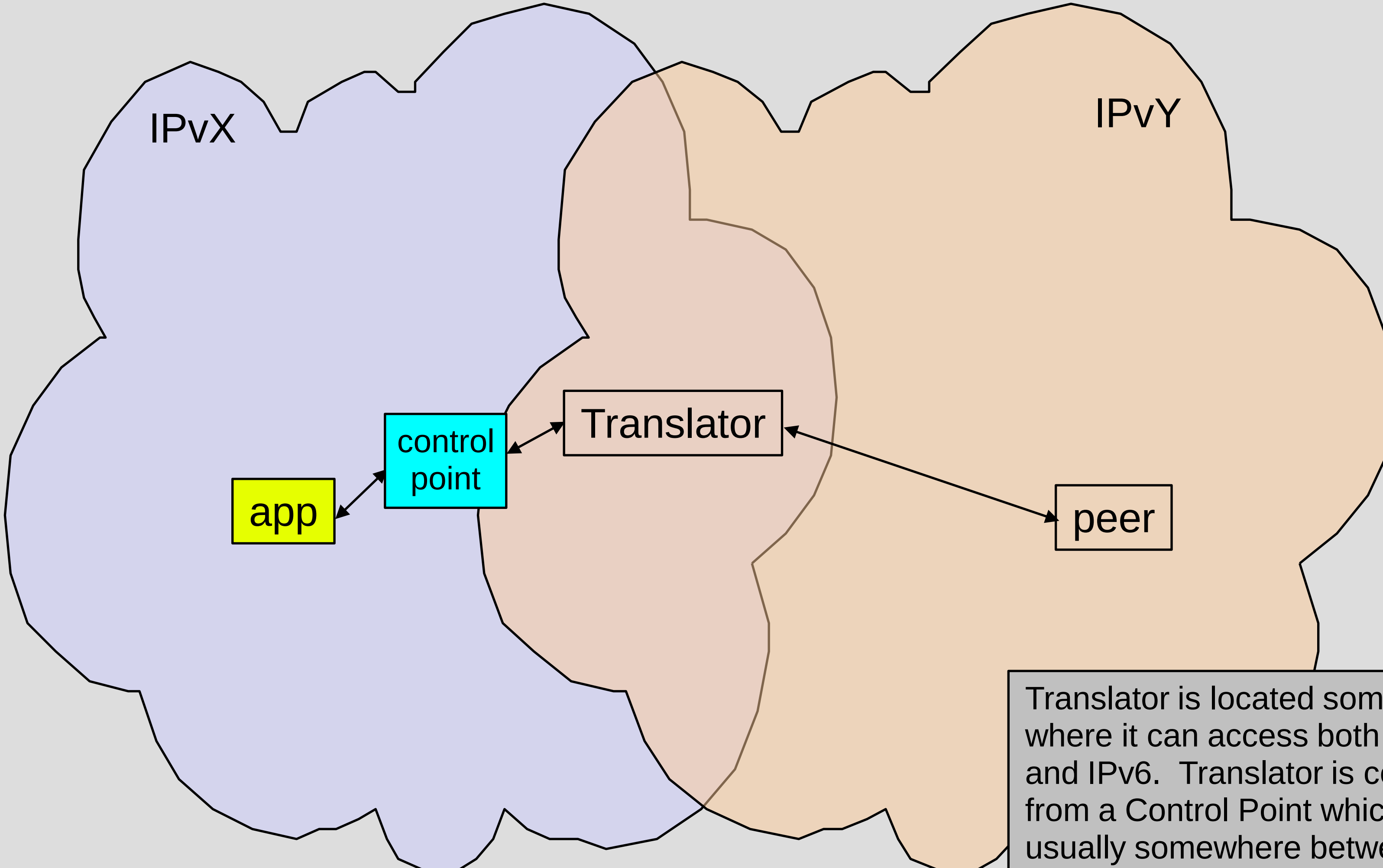
  decouple app, host, net, ISP implementation
- Provide a predictable programming model

  independent of local IPvX support or NAT configuration
- Accommodate legacy apps, hosts, nets

  without breaking DS apps
- Encourage a desirable end-state

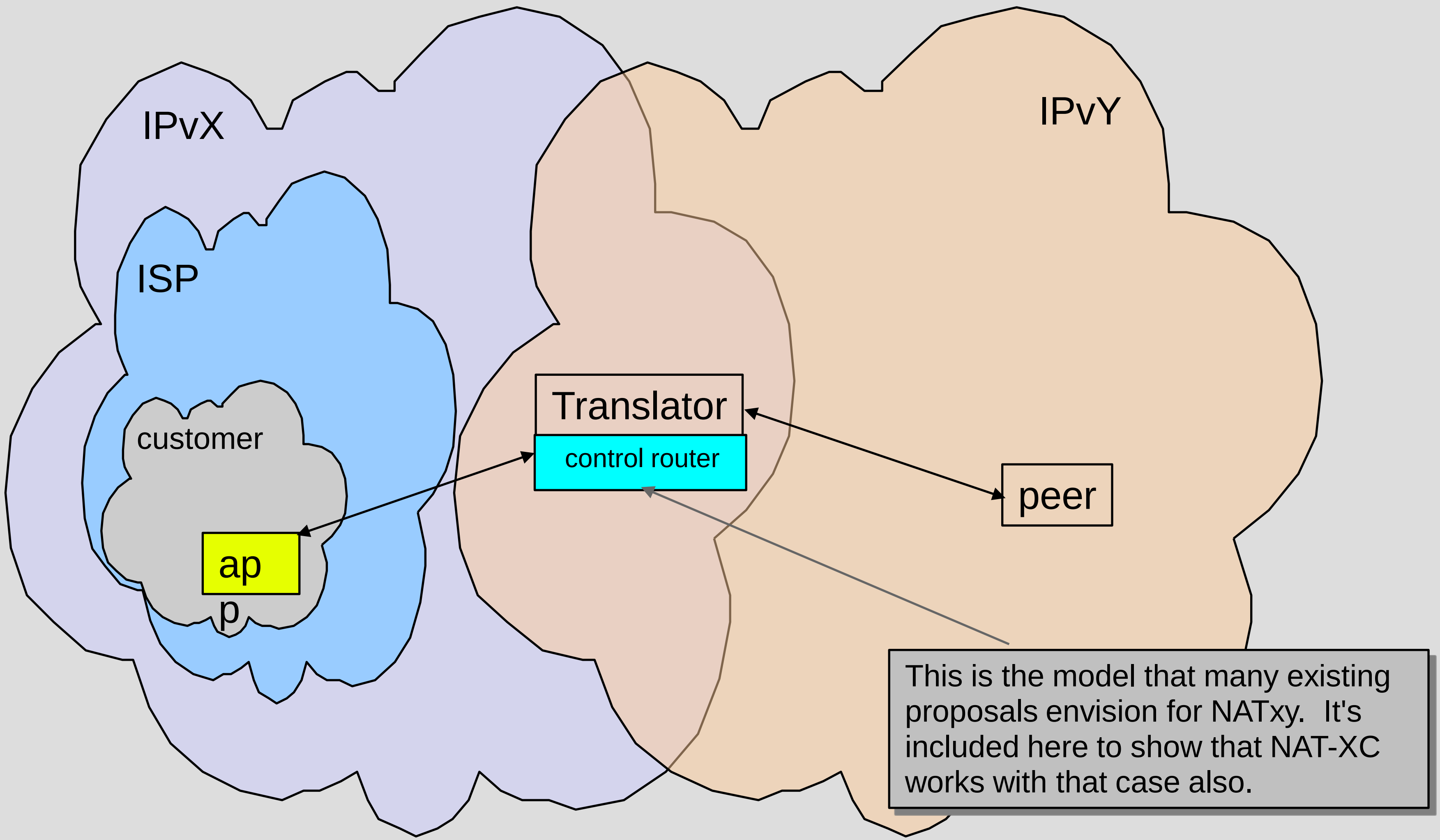
  Everything can use IPv6 everywhere
- “Make the Internet safe for applications”

Basic NAT-XC Architecture

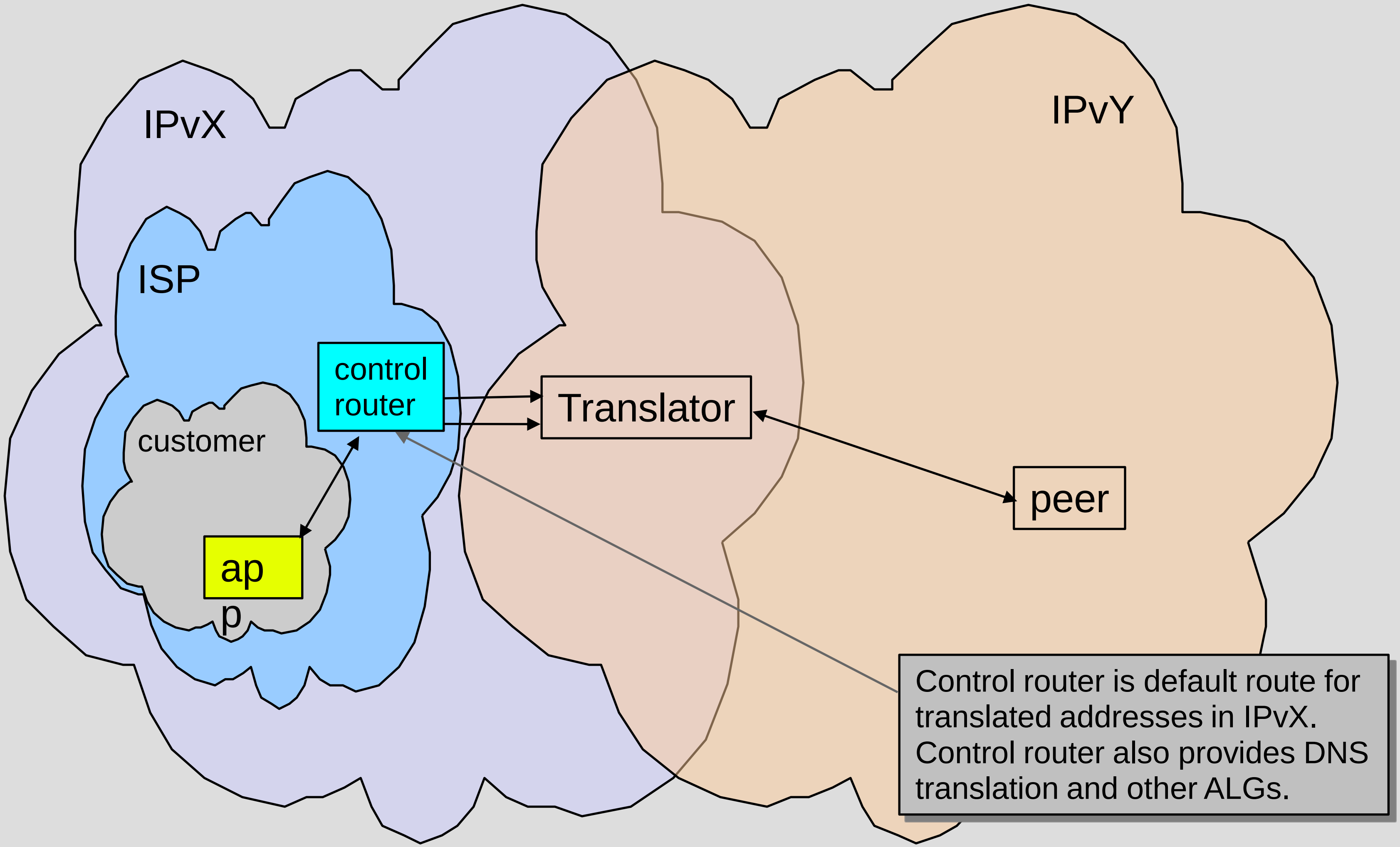


Translator is located somewhere where it can access both IPv4 and IPv6. Translator is controlled from a Control Point which is usually somewhere between the app and Translator.

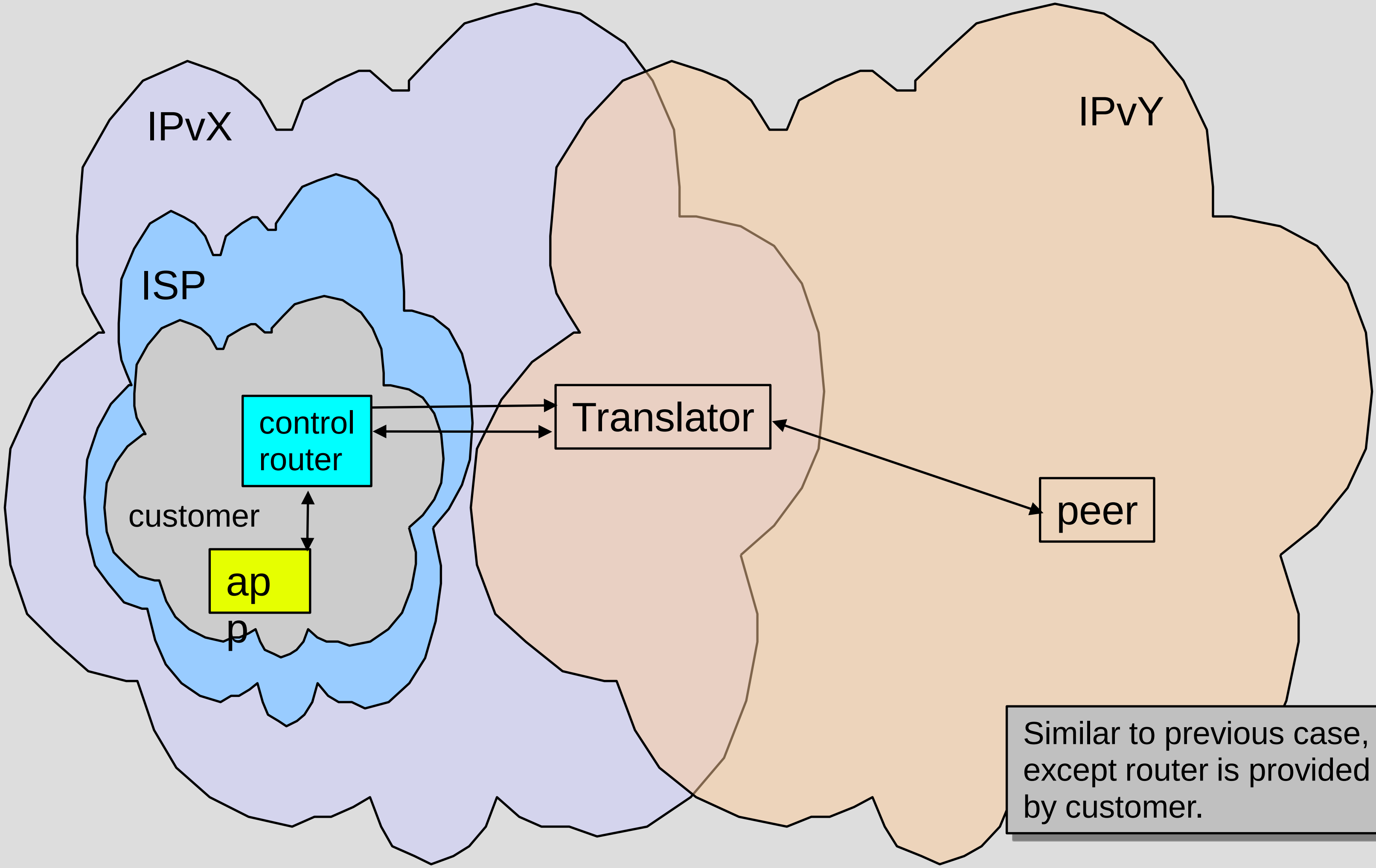
Case 0: Translator colocated with control router



Case 1: Translator controlled by ISP router

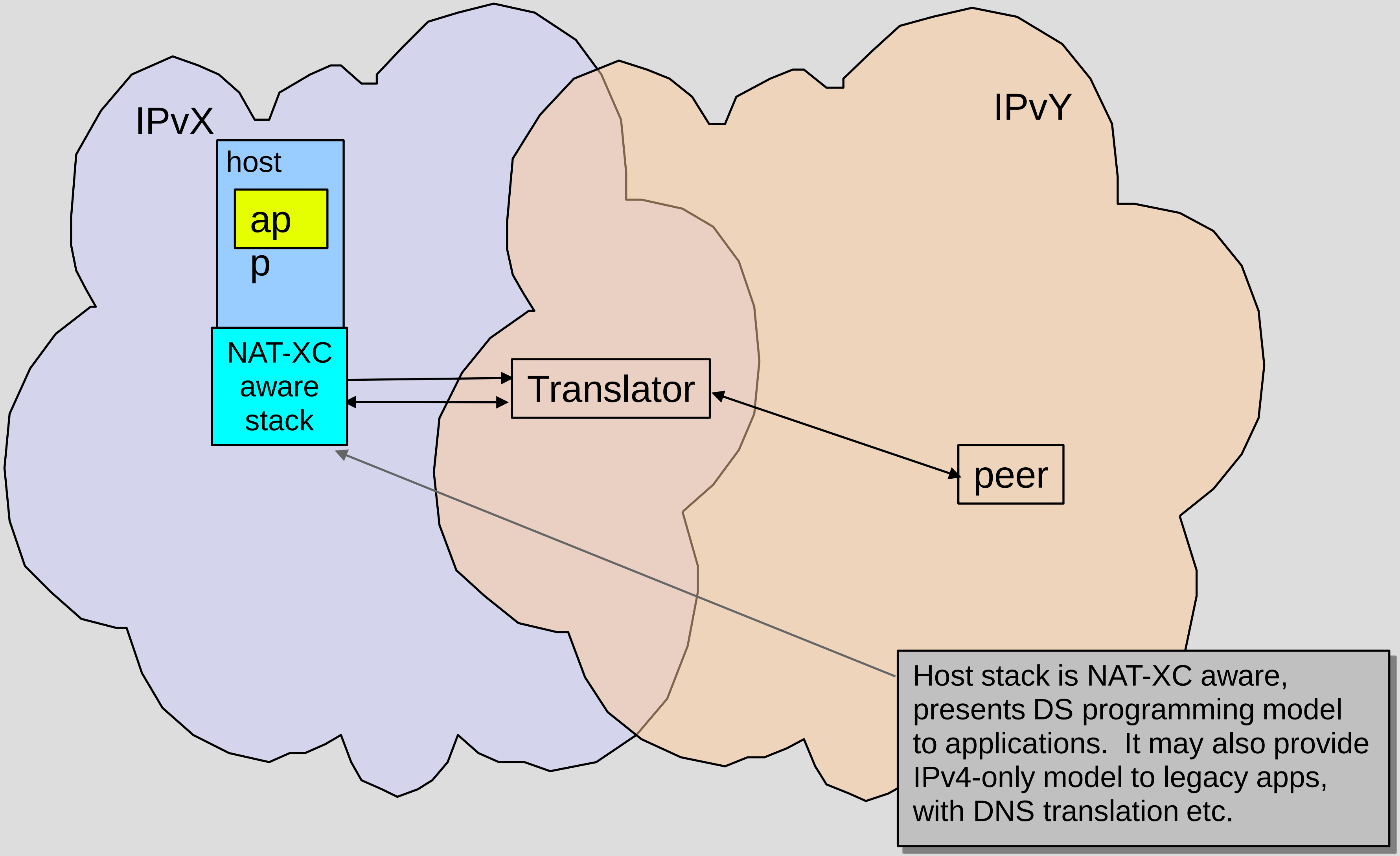


Case 2: Translator controlled by customer router

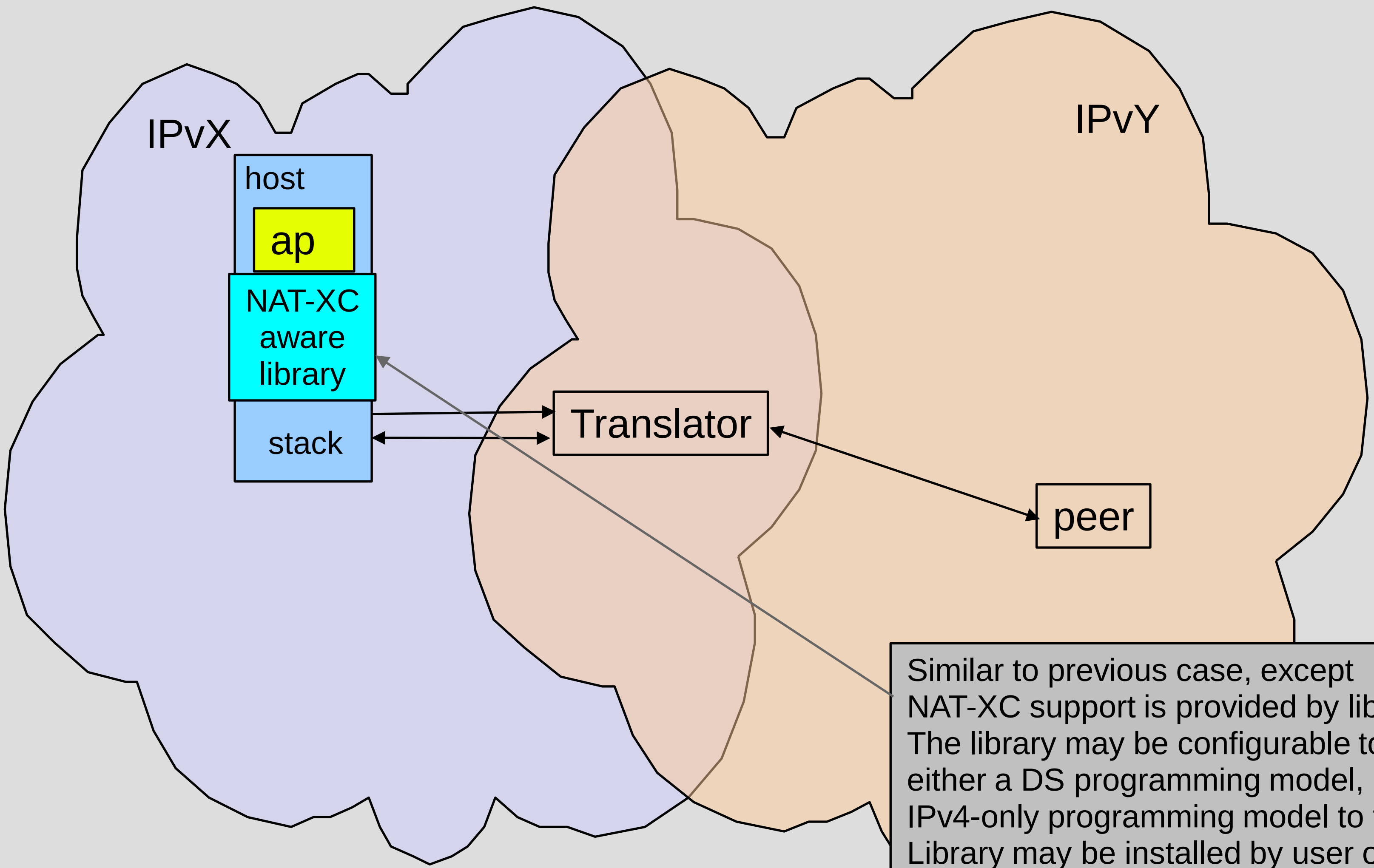


Similar to previous case, except router is provided by customer.

Case 3: Translator controlled by host stack

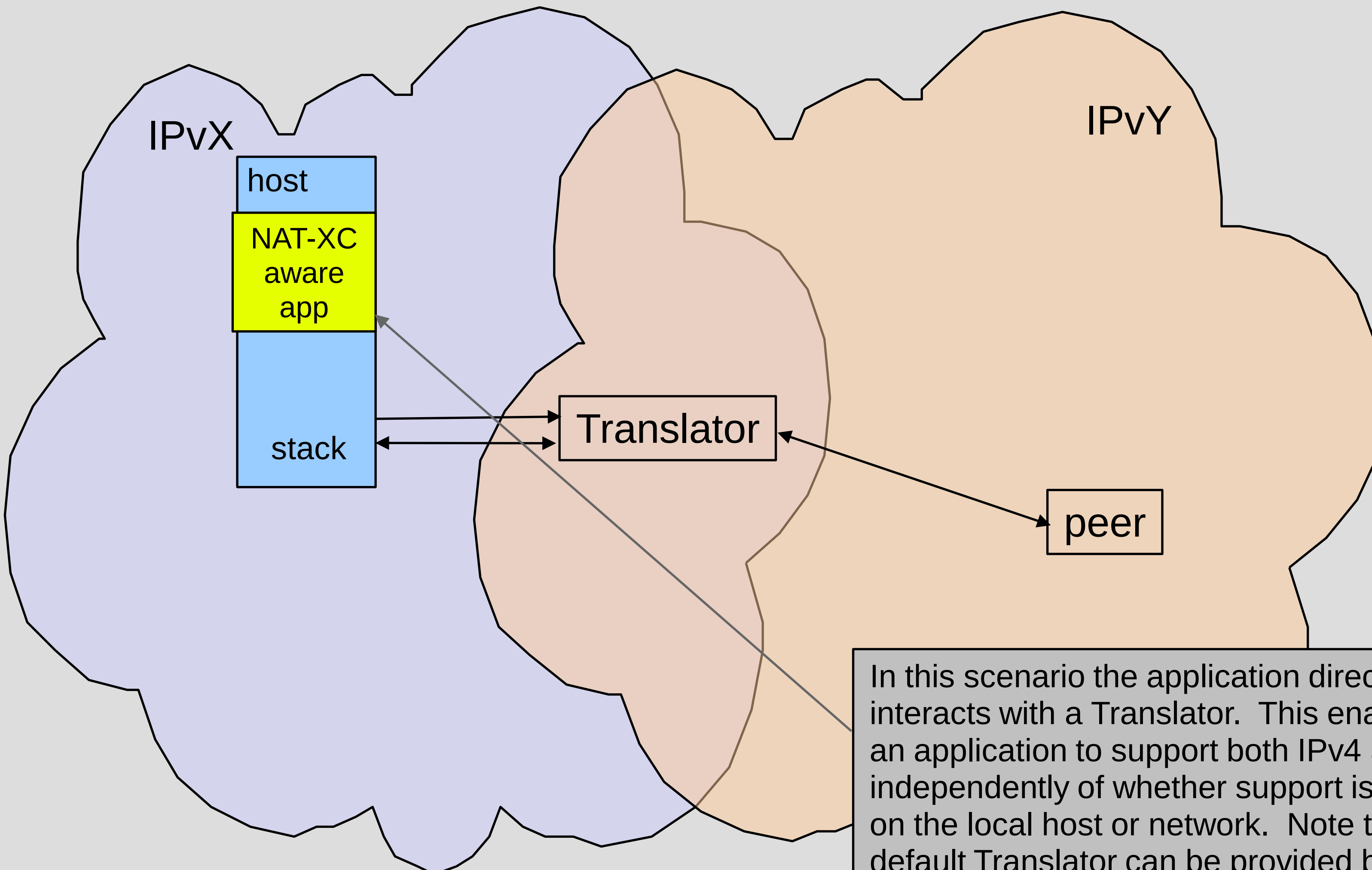


Case 4: Translator controlled by shared library / DLL



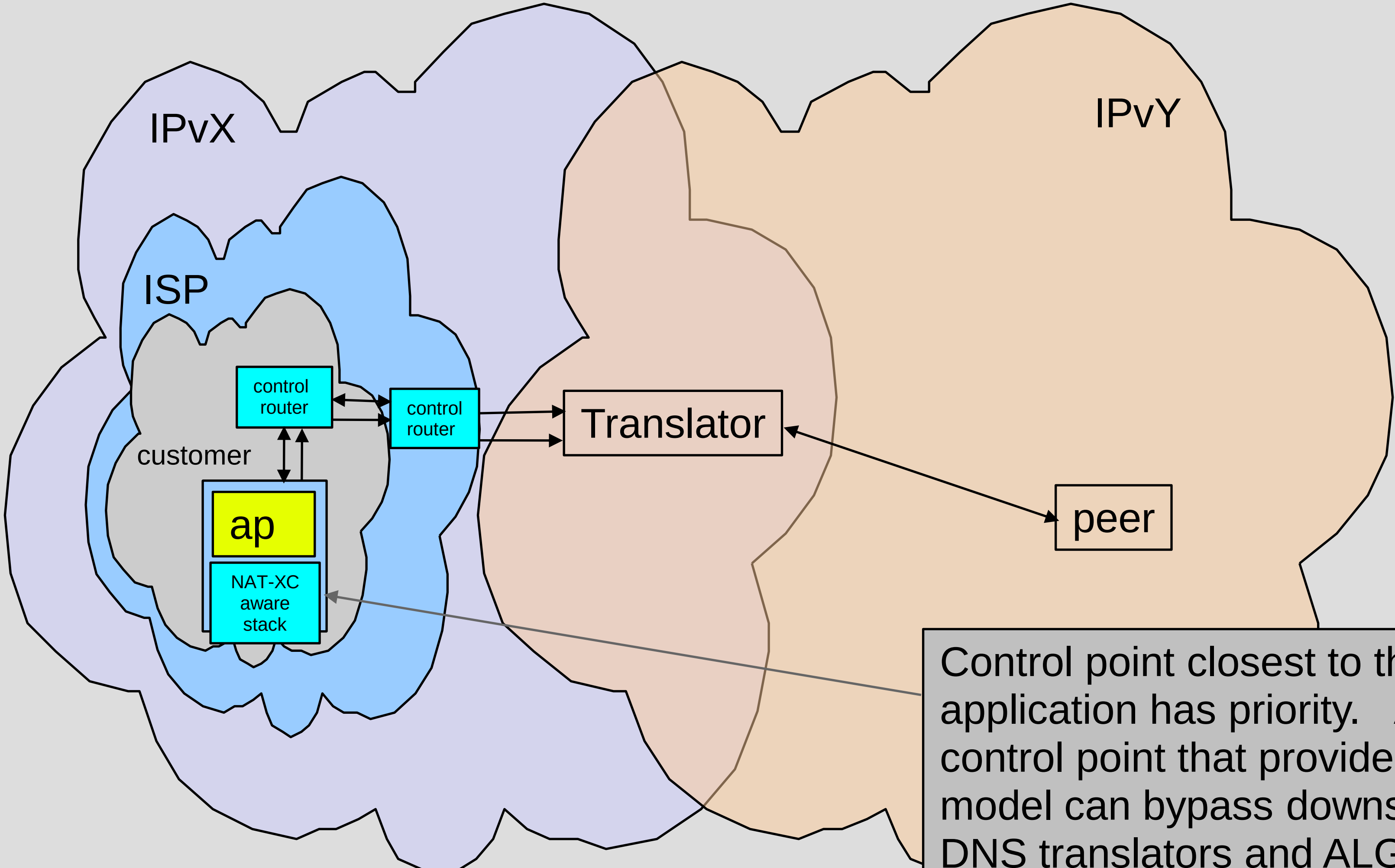
Similar to previous case, except NAT-XC support is provided by library. The library may be configurable to provide either a DS programming model, or an IPv4-only programming model to the app. Library may be installed by user or admin to enable a legacy app to adapt to NAT-XC.

Case 5: Translator controlled by application



In this scenario the application directly interacts with a Translator. This enables an application to support both IPv4 and IPv6 independently of whether support is available on the local host or network. Note that a default Translator can be provided by the application vendor and optionally overridden by the host administrator.

Case 6: Multiple control points



Control point closest to the application has priority. A control point that provides a DS model can bypass downstream DNS translators and ALGs.

What the protocol looks like

- Based on STUN
  - allow legacy NAT between CP and Translator
- Well-known “anycast” control address/port
  - one for IPv4, one for IPv6
  - can be overridden with manual configuration
- Authentication
  - control access to specific addresses, ports
  - thwart packet laundering
- Multiple bindings associated with a CP grouped together for “lease renewal”

NAT-XC protocol

- **`CreateBinding`** `(PrivateClientTA, RemoteTranslatorTA, [PeerTA,] [PiggybackPkt,] [options])`
  - remote address or port can be “wildcard” to allow the translator to assign any address/port
  - peer transport address not specified -> binding allows incoming traffic from any peer
  - client port “wildcard” -> requests entire address
- **`RenewLease`** `()`
- **`DeleteBinding`** `(RequestedTTL)`
- **`GetBindingList`** `()`
- **`BindingNotification`** messages

Protocol Usage

- To establish outgoing connection:
  - control router: triggered by new flow, or DNS
  - API: triggered by connect() or sendto() call
  - CreateBinding() from client to peer address
  - packet that triggers binding can be piggybacked
- To listen for incoming connection:
  - control router: binding explicitly configured, or requested by authenticated 3rd party
  - API: triggered by listen() call
  - API binds to local TA where it wants to listen
  - makes CreateBinding() request from that TA

NAT-XC as uniform interface to different kinds of NAT

- Neither the application nor the control point cares about the translation algorithm
  - stateless or stateful (or hybrid)
    - optimization: stateful Translator could disclose its mapping algorithm in CreateBinding response
  - doesn't care about WK vs. LIR prefix
  - port-restricted or not
  - endpoint dependence?
    (binding specific to a remote peer address)
- Permits a variety of Translator configurations
  - e.g. NAT/CPE, CGN, 3rd party service
- Generalizable to v4/v4 and v6/v6 also

NAT-XC deployment

- To use NAT-XC you need:
  - (a) Translator; (b) Control Point
- Translator:
  - ISP might supply for “free” or for cost (for v6-only or to lure customers away from v4)
  - net with both v4/v6 access can provide locally
  - 3rd party (for cost)
  - app developer can arrange for a default one
- Control Point:
  - user: upgrade OS, or install shared lib/DLL
  - network admin: install control router
  - ISP (see above)
  - app vendor: ship with app

Conclusions

- NAT-XC
  - Accommodates a variety of NAT types
    - state-less/ful, address-sharing, endpoint-specific?
  - Avoids explicit configuration of hosts, DNS translators to know mapping algorithm
  - Accommodates a variety of app types
    - apps written to DS model vs. legacy v4 model
    - “simple” (client/server) vs. “clever” (p2p) apps
  - Gives apps a uniform programming environment
  - Decouples developer/user/network/ISP constraints, that hinder deployment of IPv6
    - anyone can arrange for his apps to have DS access
  - **Costs borne by those who benefit**