Application Level Control of Ports in a Service Provider NAT environment

Dave Thaler
Dan Wing
Alain Durand
Port Control Protocol

• Service Provider NATs have problems:
  – Lack of control of port reservation / port forwarding
  – Some legacy applications will break
• A+P was one approach to address those issues
• PCP is another approach to give back control to the customers via their applications.
  – Enable applications to dynamically negotiate ports with the service provider NAT
  – Provide some level of backward compatibility with existing APIs (UPnP/NAT-PMP)
Port-Forwarding APIs

Dave Thaler
dthaler@microsoft.com
Model

• No change to IP model:
  – A full IP address is still assigned to every interface, including on NATs

• App/framework wants to learn the (full) IP address of another machine’s (the NAT’s) interface, and a port that machine will forward
  – Can’t be done using normal IP address APIs without changing the IP model
  – App/framework can then advertise in app-specific manner (SRV record, email, DHT, etc.)

• Hence this is opt-in for an app or framework
Two separate app scenarios

• Manage static port mapping
  – Management style application wants to configure a given external port to be permanently forwarded to a given port on a given machine

• Manage dynamic port mapping
  – Runtime application wants to get an external port allocated and forwarded to its port on its machine for some duration
NATUPnP Library
(Windows)

NATUPNPLib.UPnPNATClass upnpnat = new
NATUPNPLib.UPnPNATClass();
NATUPNPLib.IStaticPortMappingCollection mappings = upnpnat.StaticPortMappingCollection;

err = mappings.Add(8080,  // External port
    "TCP",  // Protocol
    80,  // Internal port
    "192.168.1.100", // Internal IP
    true,  // Enabled
    "Local Web Server"); // Description

• External port=0 means wildcard, but many NATs don’t support
NATUPnP API Observations

• Either requested port is allocated or call fails
• Internal IP parameter allows for management applications
• Only supports static port mapping (no lifetime)
  – UPnP protocol allows lifetimes, but NATs may not support them
• Interface can be determined based on internal IP parameter
DNSServiceNAT (Apple)

DNSServiceRef sdRef;

err = DNSServiceNATPortMappingCreate(&sdRef, 0, 0, kDNSServiceProtocol_TCP, htons(80), htons(8080), 3600, callBack, NULL);

• External port=0 means wildcard
DNSServiceNAT Observations

- Lifetime parameter allows for runtime applications
- External port is just a preference, it may succeed and return something else
- Lack of internal IP parameter means not designed for arbitrary management app
Port Control Protocol

draft-wing-softmaxwire-port-control-protocol-01

IETF77, March 2010

Dan Wing, dwing@cisco.com
Reinaldo Penno, rpenno@juniper.net
Mohamed Boucadair, mohamed.boucadair@orange-ftgroup.com
Port Control Protocol

• Need to offer port forwarding capability when Service Provider NAT are deployed
  – Ability to offer similar service features as per current CPE model

• Need to delegate port numbers to requesting applications/hosts to avoid enforcing ALGs at the Provider NAT
  – Overall performance of the Provider NAT not altered
PCP Requirements

• Support Large Scale NATs
  – Spanning many subscribers
• Allow subscriber apps to open ports
• IPv6
• Simple, lightweight
  – Application, proxying in CPE, and server
• Discover and control LSN
  – Without interfering with intermediate infrastructure
Why Not My Favorite Protocol?
(MIDCOM, UPnP IGD, NAT-PMP, DHCP …)

• None meet all requirements
PCP Applicability

• IPv4 address sharing
  – No NAT44 (fixed port range)
  – Stateful NAT44 (e.g., DS-Lite, LSN)
  – Stateless NAT64/NAT46
  – Stateful NAT64/NAT46

• IPv6 Simple CPE Security
PCP Basics

• Lightweight
  – Designed for deployment at large scale
  – Does not require heavy treatment at the Server side
    • Quick convergent Request/answer model
  – No permanent sessions are required to be maintained between the Client and the Server

• A subscriber can only open pinholes for his own devices
  – PCP isn’t needed in every internal server
  – E.g., Customer Premise router can open pinhole for webcam or TiVo
PCP and IPv6

• NAT64
  – Open ports for incoming IPv4 traffic
    • E.g., IPv6 HTTP server in the home accessed from IPv4 Internet

• draft-ietf-v6ops-cpe-simple-security-09
  – Open pinholes in IPv6 CPE
Client Models
PCP Client Model: UPnP IGD Proxy

- Proxies UPnP IGD to PCP
- Provides compatibility for UPnP IGD
- Applications which want specific port will likely get an error
  – Can’t help that
PCP Client Model: NAT-PMP Proxy

- Proxies NAT-PMP to PCP
- Provides compatibility for UPnP IGD
- No loss of semantics
PCP Client Model: HTTP

- Subscriber manages their own port forwarding
  - Similar to http://192.168.1.1, login as “admin”
  - Instructions at http://www.portforward.com
- Not for “Grandma”
PCP Client Model: PCP on host

- Application (or OS) implements PCP client
- Customer premise router does nothing
  - Does not proxy PCP
- draft-ietf-v6ops-cpe-simple-security
Server Models
PCP Server Model: Embedded

- PCP Server embedded in Service Provider’s NAT
- Similar to UPnP IGD, NAT-PMP
PCP Server Model: Separate

- PCP Server is outside of the NAT
- Allows existing NAT control protocol
Questions
PCP Server Models

PCP Client

Service Provider NAT

PCP Server

Internet

IPv6

H.248, MIDCOM, proprietary, etc.

Service Provider NAT

PCP Client

Internet

draft-wing-software-port-control-protocol 26
PCP Client Models

- UPnP IGD
  - Customer Premise Router
  - UPnP IGD proxy
  - PCP Server

- NAT-PMP
  - Customer Premise Router
  - NAT-PMP proxy
  - PCP Server

- HTTP
  - Customer Premise Router
  - HTTP managed
  - PCP Server

- PCP Client
Mapping APIs/protocols to PCP

• Apps shouldn’t have to know which case they’re in
• DNNServiceNAT API / NAT-PMP protocol maps directly
• NATUPnP (v1) API / UPnP-IGD protocol more complicated
  – It can be done successfully, but it’s kludgy