ICE and libnice

Connectivity despite NAT?

Philip Withnall

2014-10-16

What is the problem?

- IPv4 address exhaustion
- Network Address Tunnelling (NAT)
- Client-to-client connections are not end-to-end connected
- NAT can't demultiplex incoming packets to listeners

Port translation

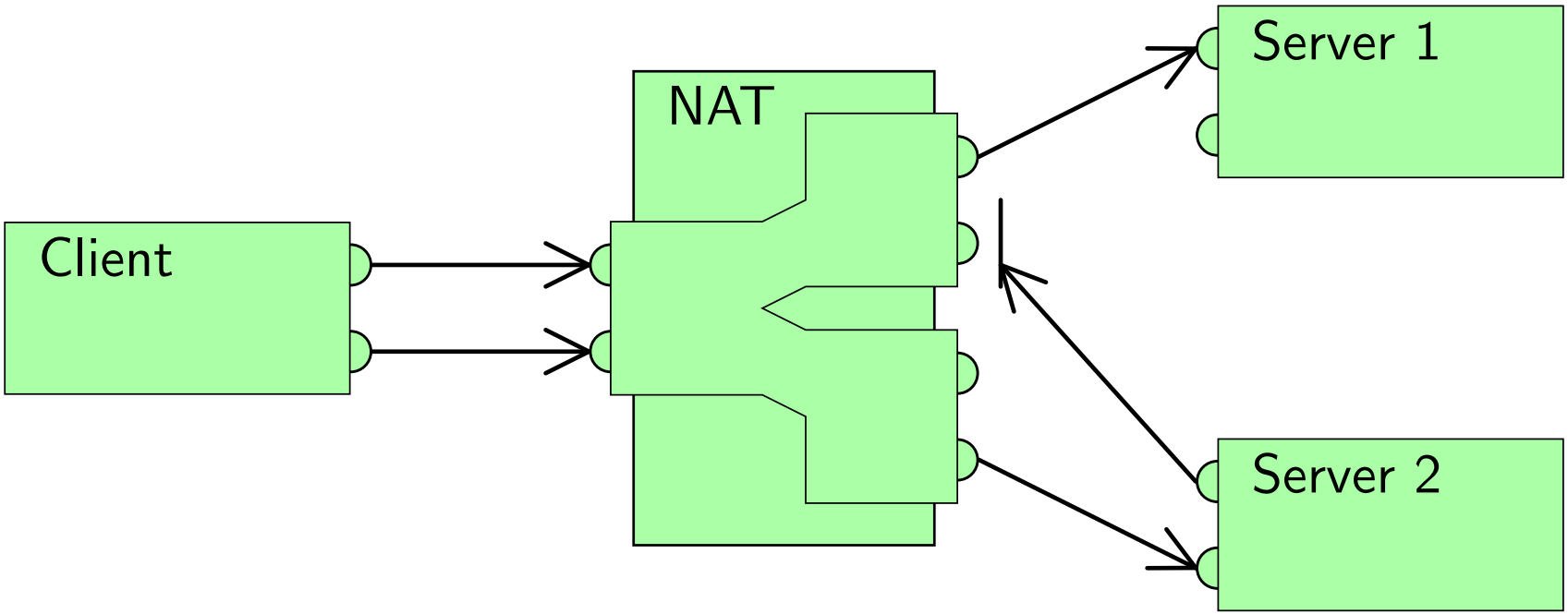

Figure: Symmetric NAT. CC-BY-SA 3.0, `http://en.wikipedia.org/wiki/File:Symmetric_NAT.svg`

Network topologies

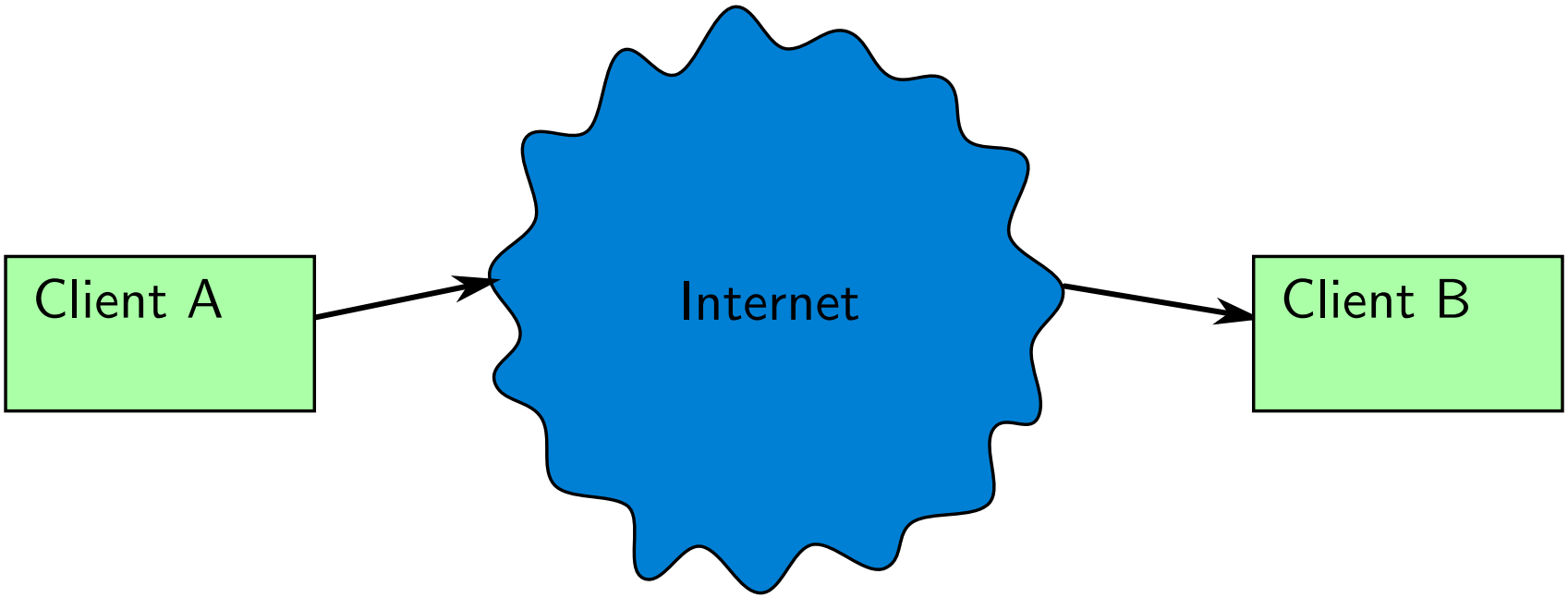

Figure: Two clients communicating without NATs.

Network topologies

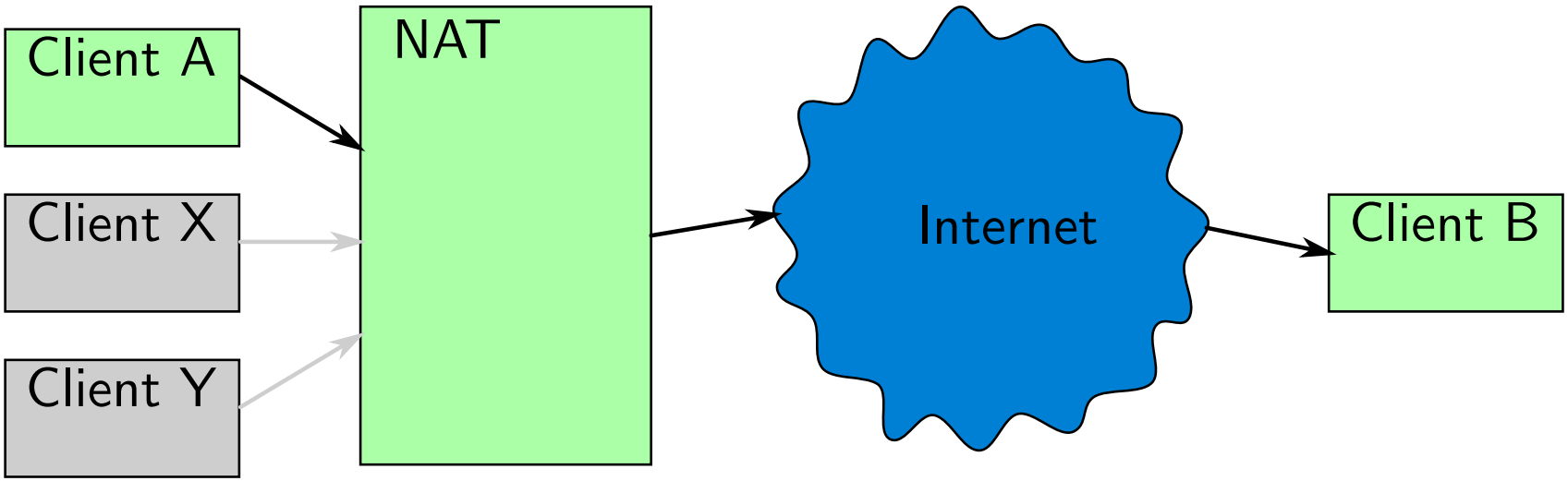

Figure: One client behind a NAT, the other not.

Network topologies

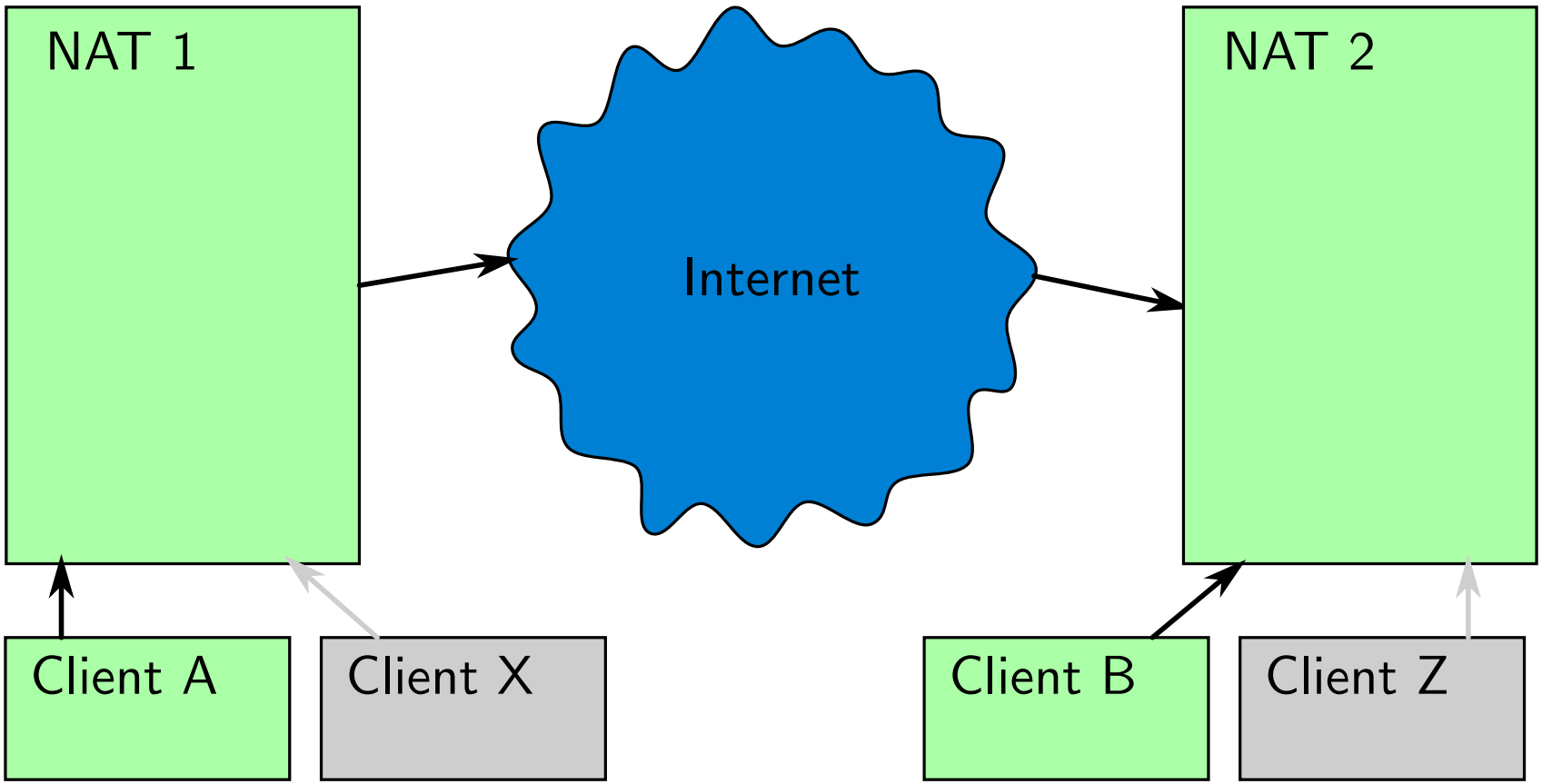

Figure: Both clients behind NATs.

Network topologies

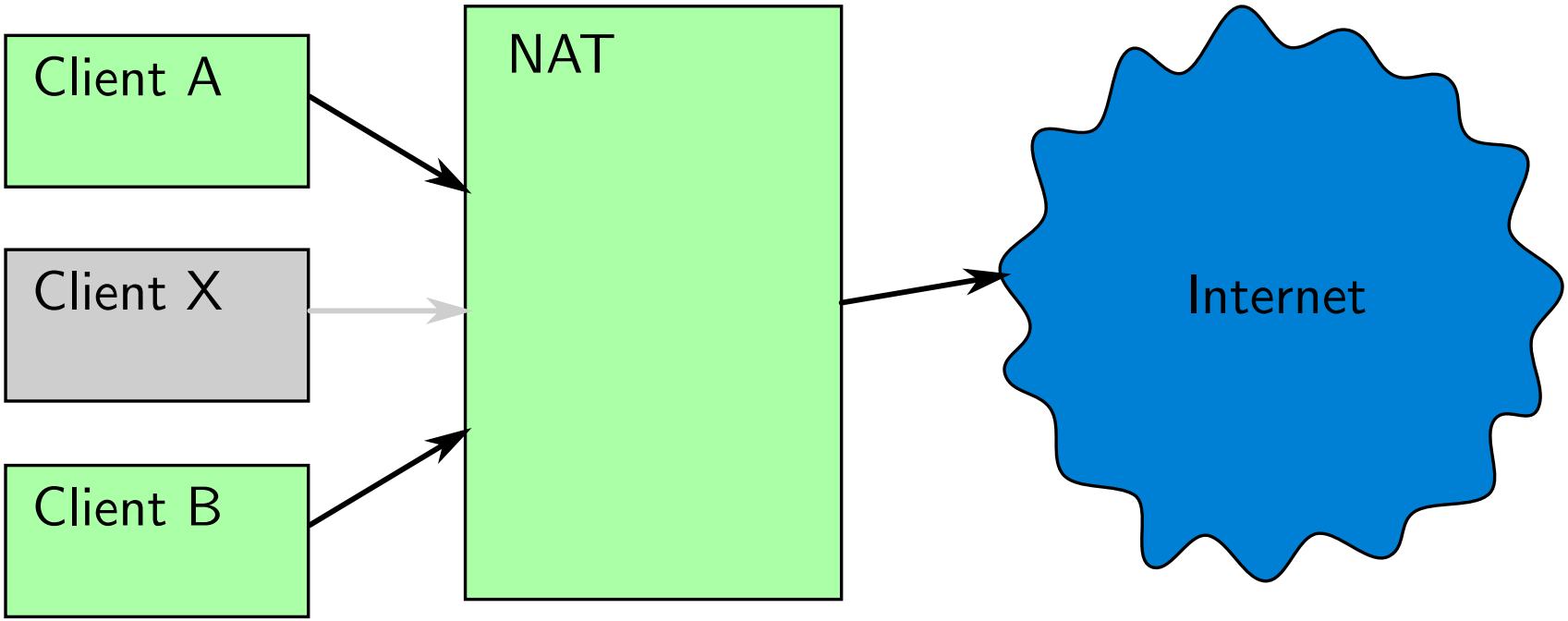

Figure: Both clients behind the same NAT.

What is the solution?

- Port forwarding
- Session Traversal Utilities for NAT (STUN)
- Traversal Using Relays around NAT (TURN)
- Interactive Connectivity Establishment (ICE)
- Hole punching

STUN

- RFC 5389 (for those who like that kind of thing)
- Client–server protocol
- Allows clients to find their WAN IP address
- Typically UDP based
- Has bells and whistles: authentication, encryption, message integrity, reliability, etc.

TURN

- RFC 5766
- Relays connections through a well-known server outside the NAT
- Expensive, slow, high latency
- Both clients are guaranteed to be able to connect to the TURN server

ICE

Basically,

$$\text{ICE} = \text{STUN} + \text{TURN}$$

ICE

- RFC 5245 (117 pages of gripping reading!)
- Clients are ignorant of network topologies and NAT configurations
- Clients have an out-of-band signalling channel
- Candidates:
  - Host
  - Server reflexive
  - Peer reflexive
  - Relayed
- Candidates are paired and checked
- Pairs are prioritised and selected

Show me the code!

```
context = g_main_context_new ();
loop = g_main_loop_new (context, FALSE);
agent = nice_agent_new (context,
                        NICE_COMPATIBILITY_RFC5245);
stream_id = nice_agent_add_stream (agent, 1);

/* Do _not_ use this in production code:
 * use nice_agent_recv_messages() instead. */
nice_agent_attach_recv (agent, stream_id, 1,
                        context, recv_cb, NULL);
nice_agent_gather_candidates (agent, stream_id);

/* Send credentials to the other client via the
 * signalling channel. */
nice_agent_get_local_credentials (agent, stream_id,
                                  &ufrag, &pwd);

g_main_loop_run (loop);
```

Show me the code!

```
static void
new_candidate_full_cb (NiceAgent *agent,
                       NiceCandidate *candidate,
                       gpointer user_data)
{
    /* Send the @candidate to the other client
     * over the signalling connection. */
}

static void
candidate_gathering_done_cb (NiceAgent *agent,
                             guint stream_id,
                             gpointer user_data)
{
    /* Notify the other client that no more
     * candidates will be sent. */
}
```

Show me the code!

```
static void
component_state_changed_cb (NiceAgent *agent,
                            guint stream_id,
                            guint component_id,
                            guint state,
                            gpointer user_data)
{
    if (state == NICE_COMPONENT_STATE_READY) {
        /* Connection established!
         * You can start sending now. */
    }
}
```

Practical example

- Client and agent
- Signalling server
- Relay server controlling multiple TURN servers

The future

- TURN-TCP support
- High-level API for candidate gathering and negotiation
- Pseudo-TCP performance improvements

Miscellany

RFC 5389 (STUN) `http://tools.ietf.org/html/rfc5389`

RFC 5766 (TURN) `http://tools.ietf.org/html/rfc5766`

RFC 5245 (ICE) `http://tools.ietf.org/html/rfc5245`

libnice `http://nice.freedesktop.org/`