# ICE

Jonathan Rosenberg
Cisco Systems

# Draft Status

- Unfortunately, draft update contained no changes, just a version number roll
- Several issues identified right after last meeting, raised a concern that things weren’t quite right yet
- Additional issues were raised from folks using the protocols about operational concerns

# Changes to be made

- Clarify re-INVITE behaviors
  - If you remove current high priority candidate, need to change username/pass on lower priority ones to force retry
- TURN back to informative
- RTCP bandwidth parameter to 00 if you are not using RTCP
- Discussion on issues of lots of STUN startup packets and impacts on congestion

# List Issue #1: Obfuscating ICE address

- Concerns about NATs that try to “help” by rewriting instances of private addresses in packets from inside to outside
- RFC3489bis deals with this by xor’ing IP address with transaction ID. Do we need something similar?
- Rough consensus was no

# List Issue #2: RTP Specificity

- Problem
  - Candidate attribute is RTP specific, and provides an alternative for a single IP/port for RTP and RTCP
  - Several cases where other addresses need candidates
    - Non-RTP transports
    - RFC 2733, where IP is transported in a=fmtp
- Propose generic candidate attribute

# Grammar

candidate-attribute = "candidate" ":" id SP qvalue SP
user-frag SP password SP
unicast-address SP port
candidate-addr SP candidate-port

v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
c=IN IP4 1.2.3.4
a=recvonly
m=audio 49170 RTP/AVP 0
a=candidate 1 0.4 adsasda 9as8dasd 1.2.3.4 49170 10.0.1.1 8700
a=candidate 2 0.4 asf9fdf8 00d-ffas 1.2.3.4 49171 10.0.1.1 8765

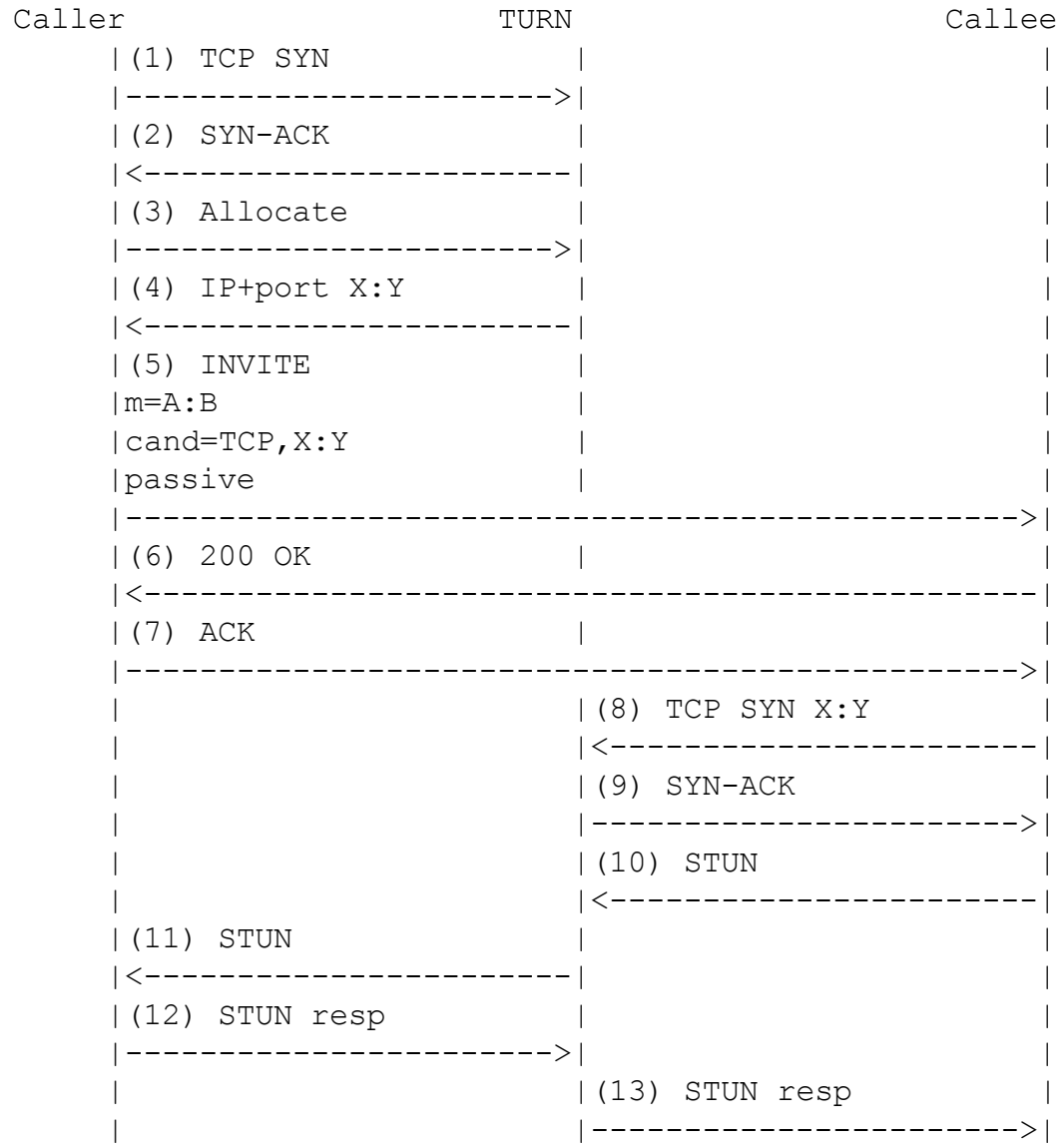
# Issue #2 continued

- May desire certain addresses to use the same group of candidates
  - i.e., RTP and RTCP should both go along same path
- This is accomplished by setting the priority identically
- Appeared to be consensus on these points. OK?

# Issue #3: TCP alternates for UDP

- Problem statement
  - For an UDP RTP stream, the only alternates you can choose are UDP
  - In the worst of environments, only outbound TCP to a server will work
  - We have defined RTCP over TCP, and TURN allows you to obtain a TCP address/port
  - We have no way to decide whether to use this worst-case option
  - We want ICE to be able to make VOIP “just work” in today’s common cases
    - And it won’t in all cases
    - Users and enterprises won’t understand why not

# Call Flow


Caller
TURN
Callee
(1) TCP SYN
(2) SYN-ACK
(3) Allocate
(4) IP+port X:Y
(5) INVITE
m=A:B
cand=TCP,X:Y
passive
(6) 200 OK
(7) ACK
(8) TCP SYN X:Y
(9) SYN-ACK
(10) STUN
(11) STUN
(12) STUN resp
(13) STUN resp

# Questions and Issues

- Does this represent a technique for circumventing the firewall policy?
  - Does blocking outbound UDP imply that VoIP service is being blocked?
    - No
  - Would there be a way to block just this?
    - Yes – block TURN ports
- Would this be required?
  - SHOULD implement, with reasons why, just like TURN
  - Policy could turn it off in an endpoint, like any other candidate technology

# Questions and Issues

- The big grouping problem
  - Concern that we may need to convey parameters for a candidate that are different from the m/c line values for the default
    - Problem for just regular candidates?
  - Concern that you can't use the defaults by just replacing "RTP" with "TCP/RTP" to identify the profile for TCP
    - RTP/AVPF exists, but not TCP/RTP/AVPF, and it might not need to
- And then it dawned on me – a solution that solves this and the other issues

# Issue #4: Figuring out what happened

- Concerns have been raised that ICE is hard to diagnose
  - Final IP/port that is used is never signaled
  - Final IP/port that is used may never have even appeared in an SDP
    - Learned through p2p STUN – used to be signaled
- Presumption is that SIP signaling is logged and that provides data for diagnosis
- Proposal made on sipping to send a re-INVITE after all done with the final choice

# Issue #5: Precondition Interactions

- If ICE is in use, when are preconditions considered met?
  - Is it assured for all candidates?
  - Is it assured for just the successful ones?
  - Is the solution specific to the precondition?

# Issue #6: Middlebox Interactions

- There are lots of things in the network that look at the SDP and open firewalls, establish QoS and do other things
  - Midcom, 3gpp PCSCF, Cablelabs PCMM
  - SBCs that modify the SDP are different (though also a concern)
- These things look at the m-line/c-line for the IP/port
  - This IP/port will be wrong
- Result: things stop working with ICE when they used to work

# Issue #7: Dynamic RTP Changes

- With ICE, the place to which RTP is sent will change dynamically as connectivity checks succeed
  - Will interact with jitter buffers
  - May make audio quality worse during the check periods

# Issue #8: Ugliness in STUN/RTP Demux

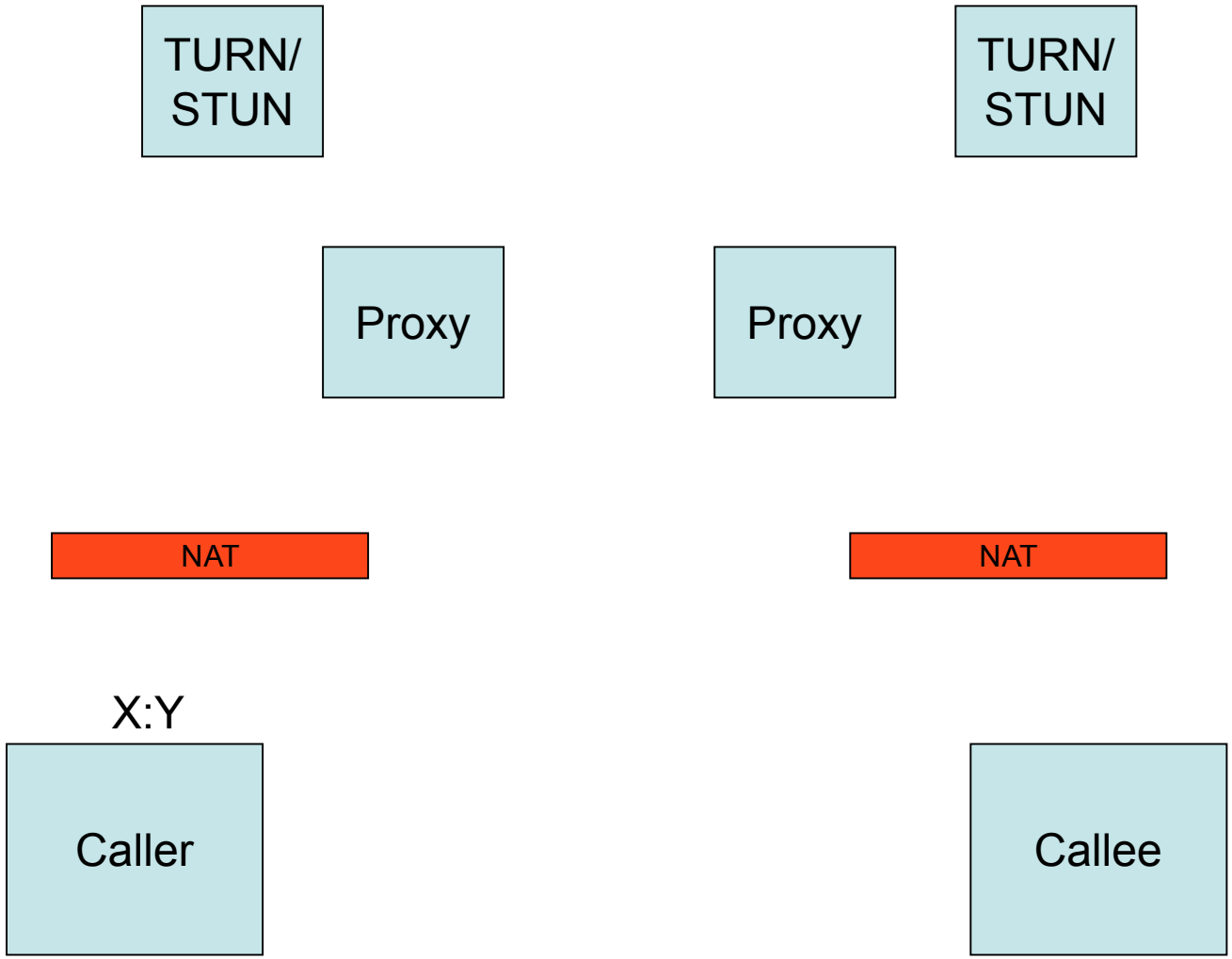
- Mechanism requires STUN/RTP demux without a clear synchronization point at which you go from one to the other
  - Has raised implementation issues
  - To avoid it, current version requires a separate local transport address for each derived one
    - However, this still problematic with forking

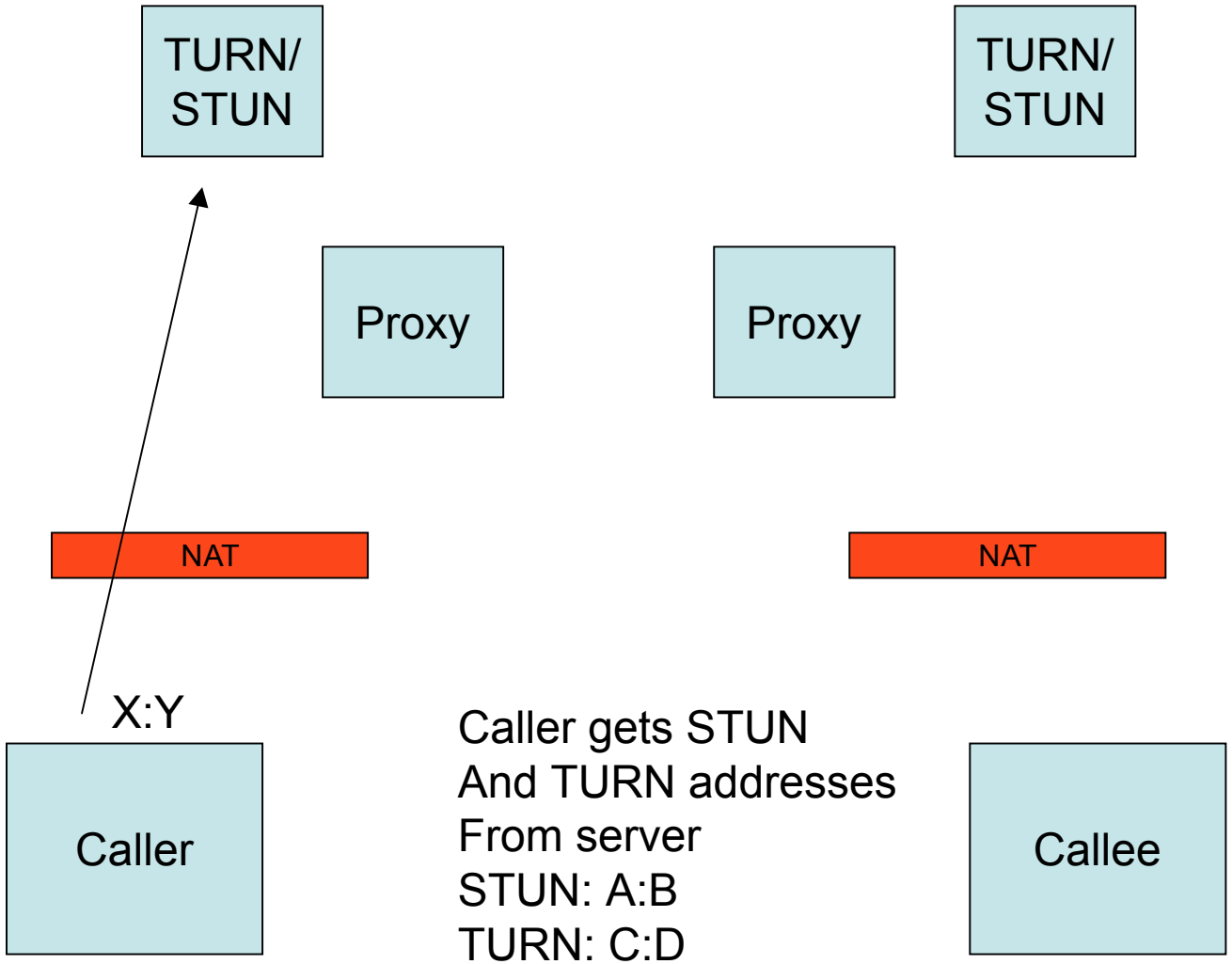
# Issue #9: SRTP Interaction

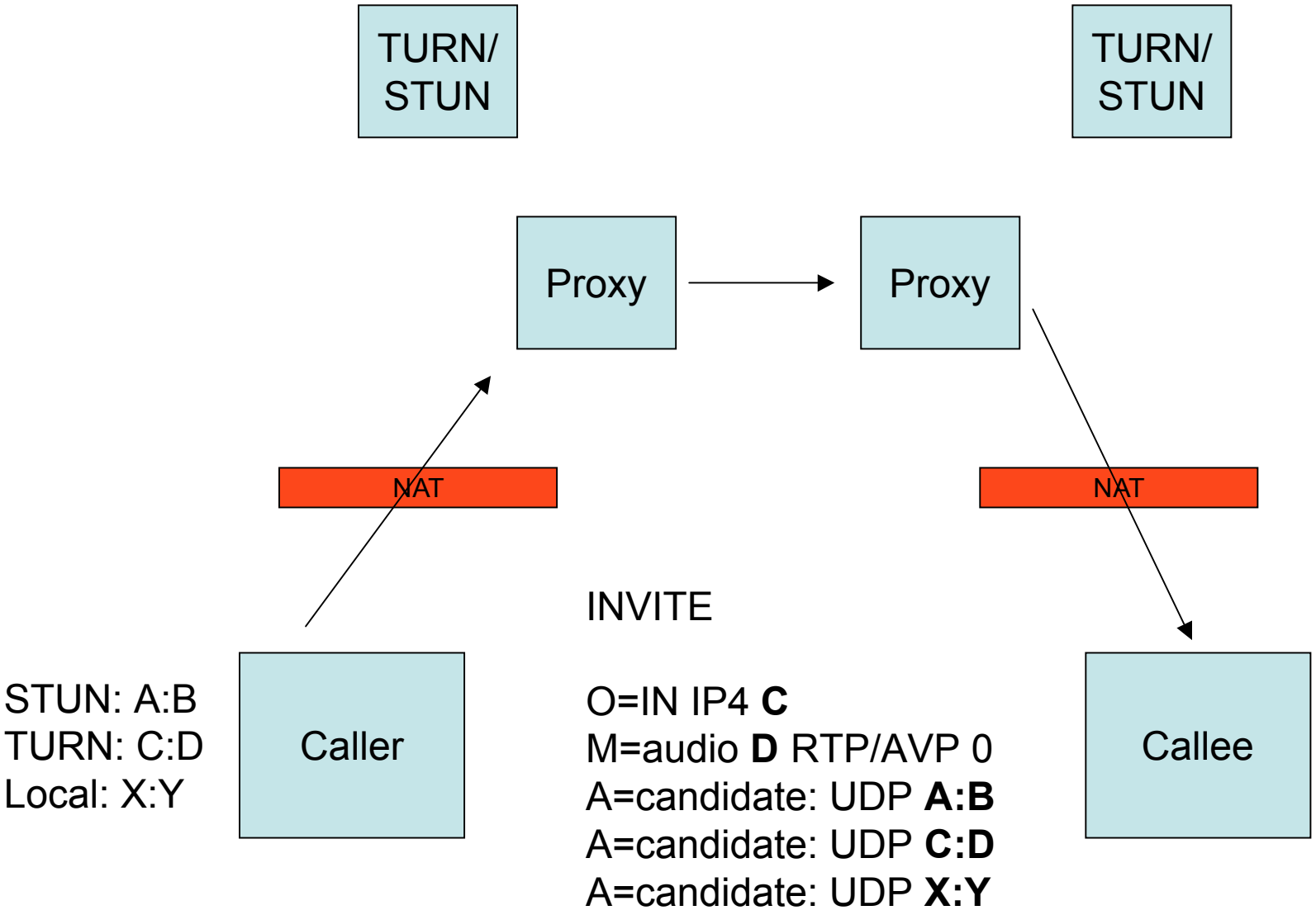
- Simply unclear what the current interaction is

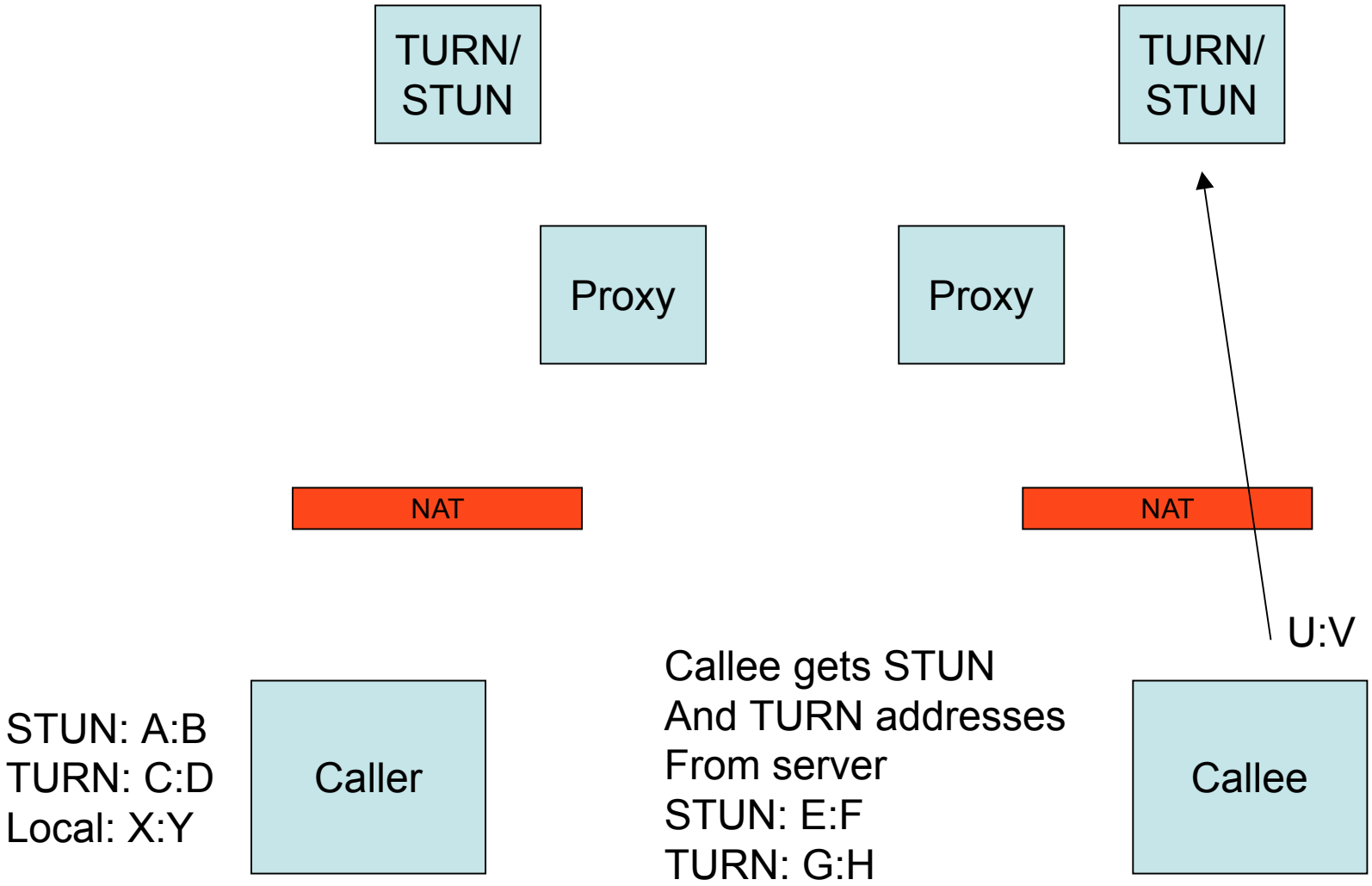
# One Solution

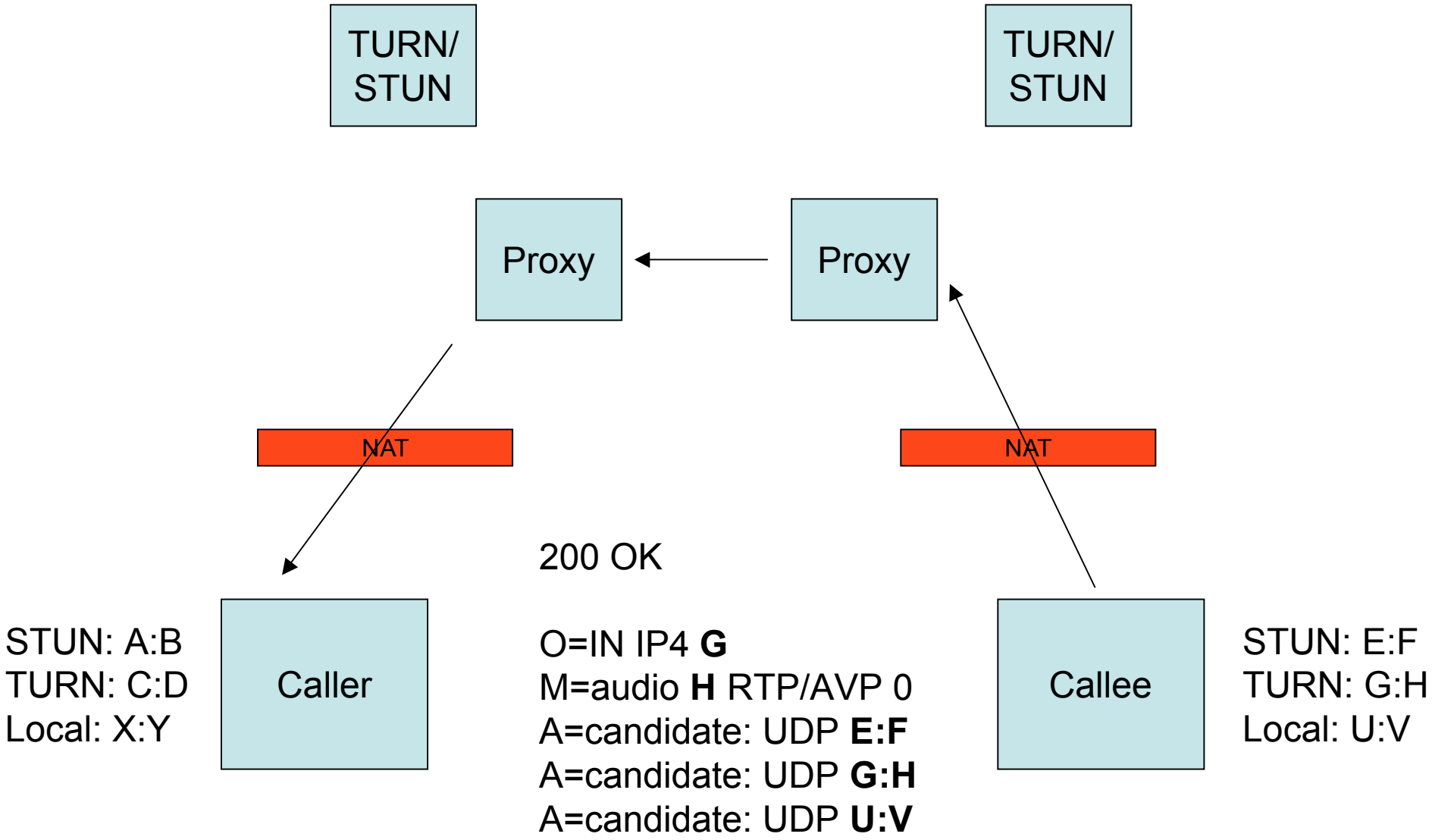
- Root cause of all of these problems is a single fact
  - The peer in the dialog starts sending media to the new address once the connectivity check succeeds
- Proposal: separate these
  - Always send media to the IP/port in the m/c lines
  - Only send connectivity checks to the IP/ports in the candidate attributes
  - When connectivity checks succeed, and it is determined that there is a desire to change where media is received, do a re-INVITE or UPDATE that “promotes” the IP/port from a candidate to the m/c line
  - Note: this does not increase call setup time or PDD!

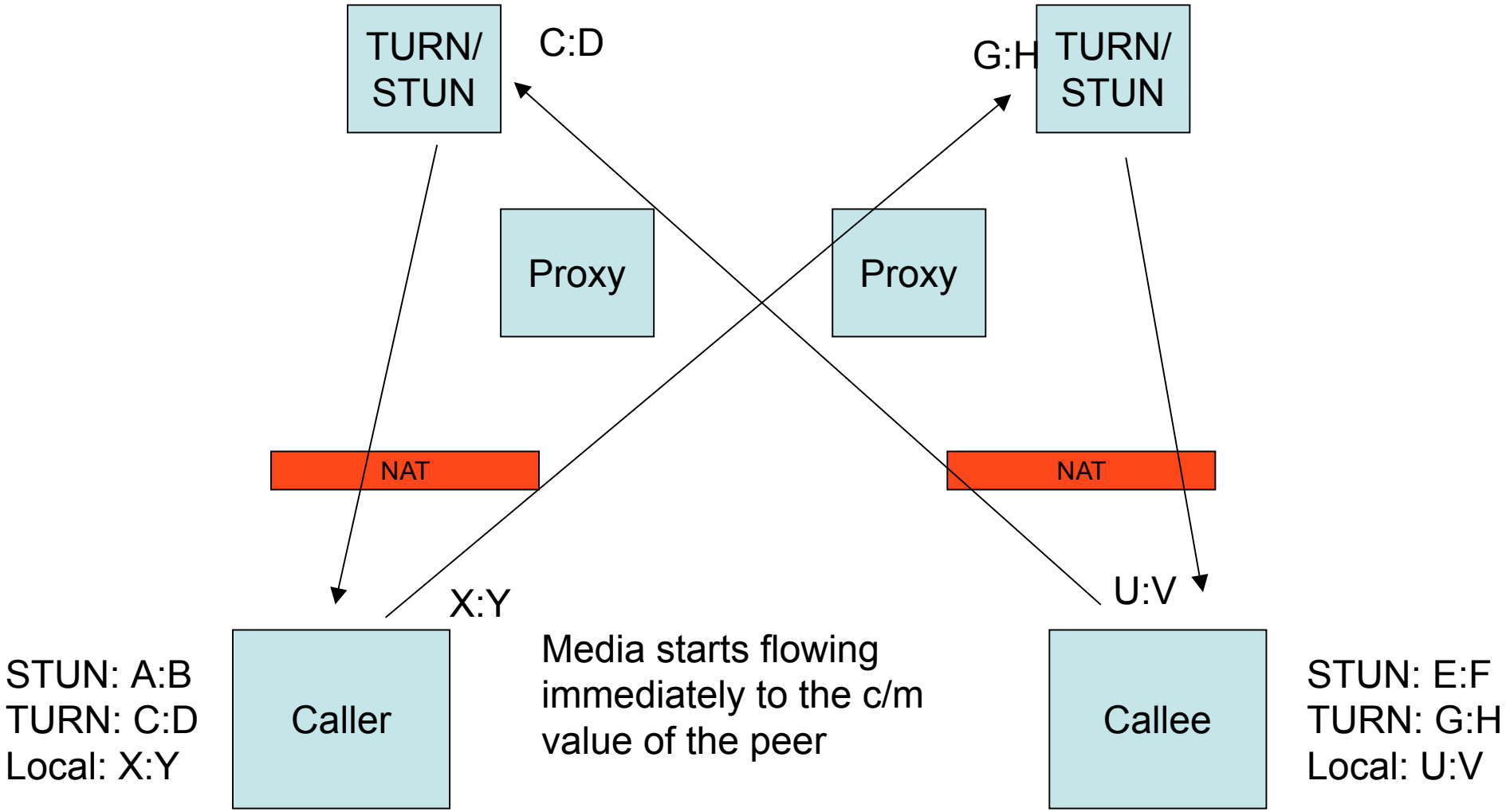

TURN/
STUN
TURN/
STUN
Proxy
Proxy
NAT
NAT
X:Y
Caller
Callee

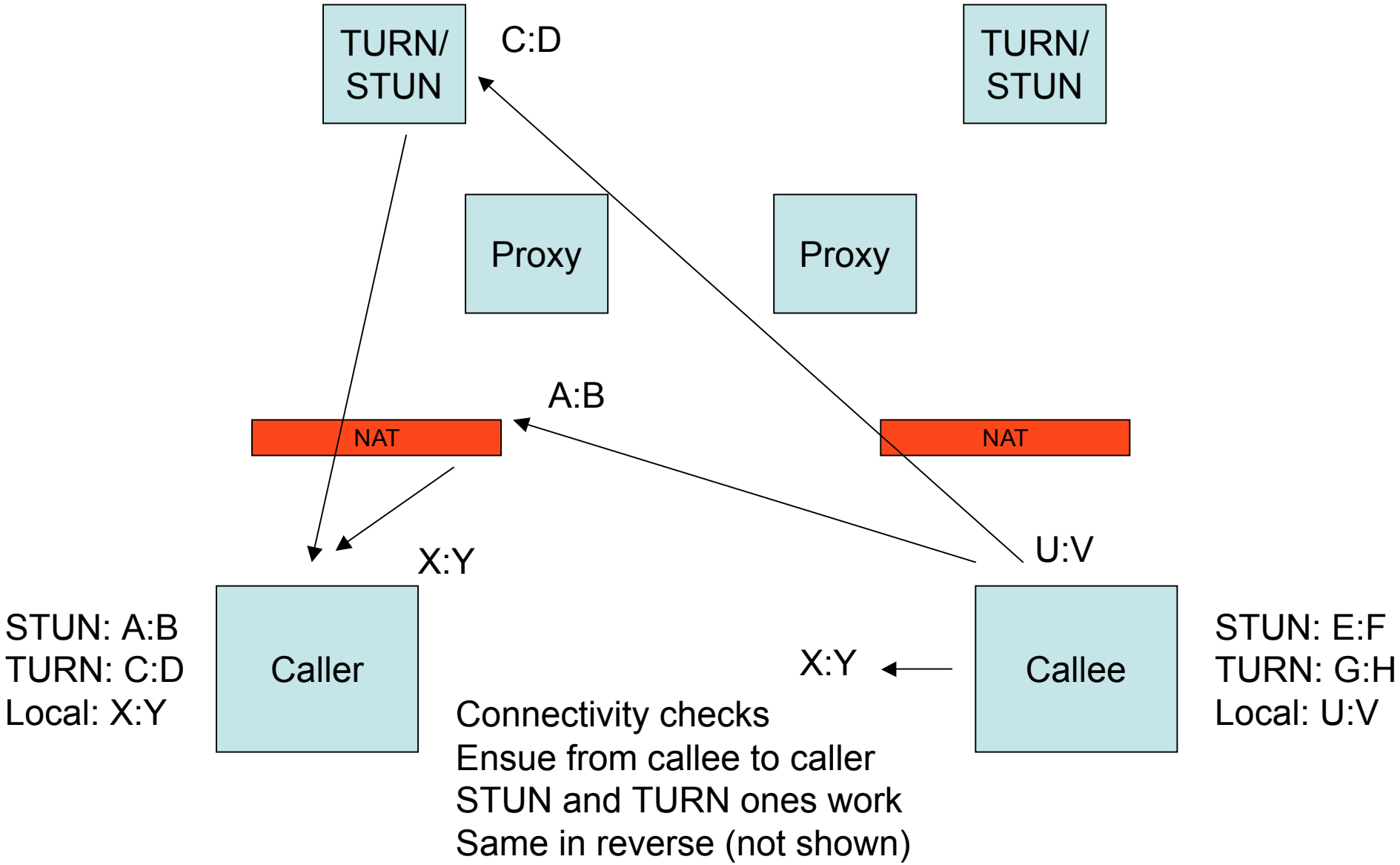

TURN/
STUN
TURN/
STUN
Proxy
Proxy
NAT
NAT
X:Y
Caller
Caller gets STUN
And TURN addresses
From server
STUN: A:B
TURN: C:D
Callee

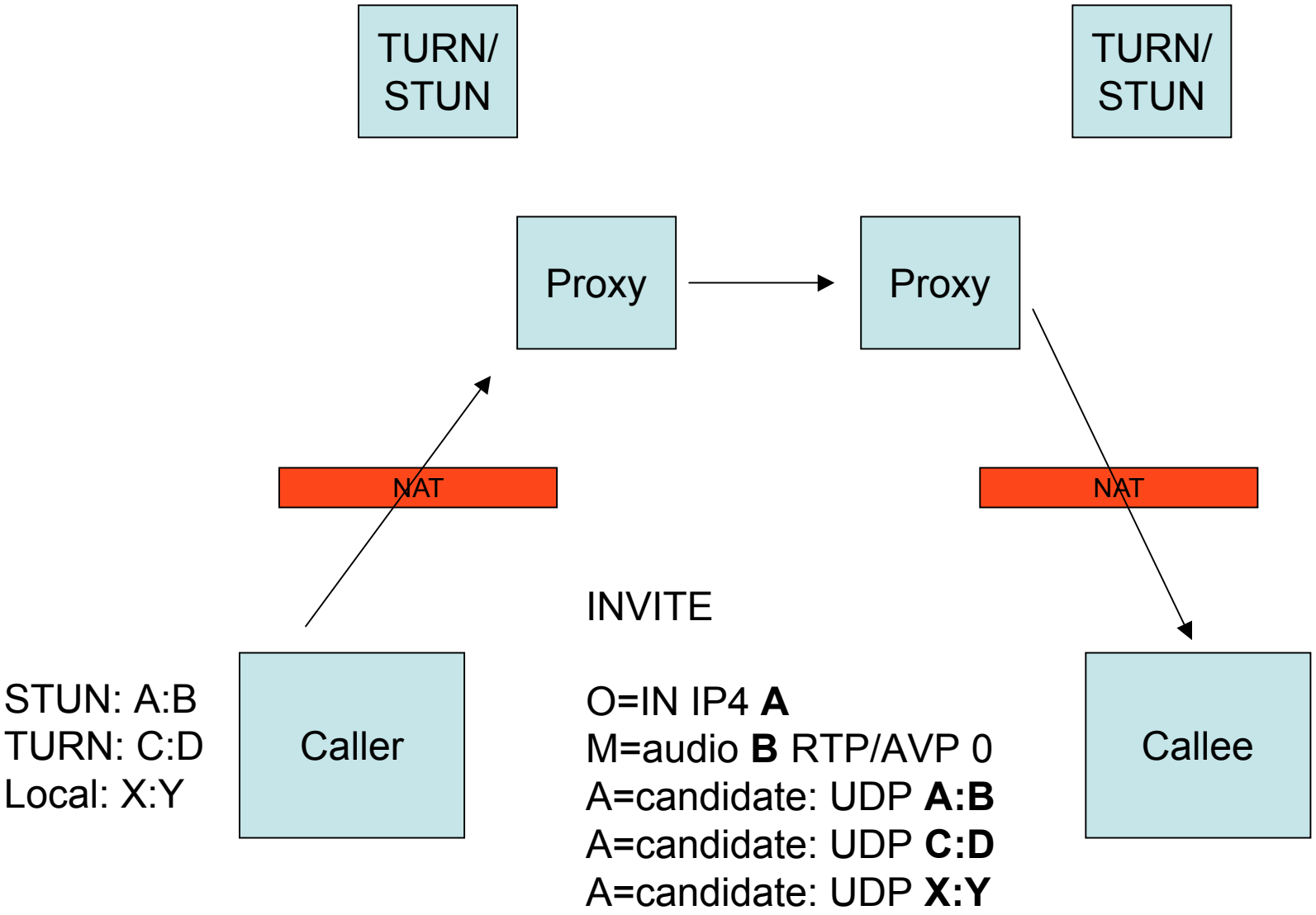

TURN/
STUN
TURN/
STUN
Proxy
Proxy
NAT
NAT
INVITE
STUN: A:B
TURN: C:D
Local: X:Y
Caller
O=IN IP4 **C**
M=audio **D** RTP/AVP 0
A=candidate: UDP **A:B**
A=candidate: UDP **C:D**
A=candidate: UDP **X:Y**
Callee

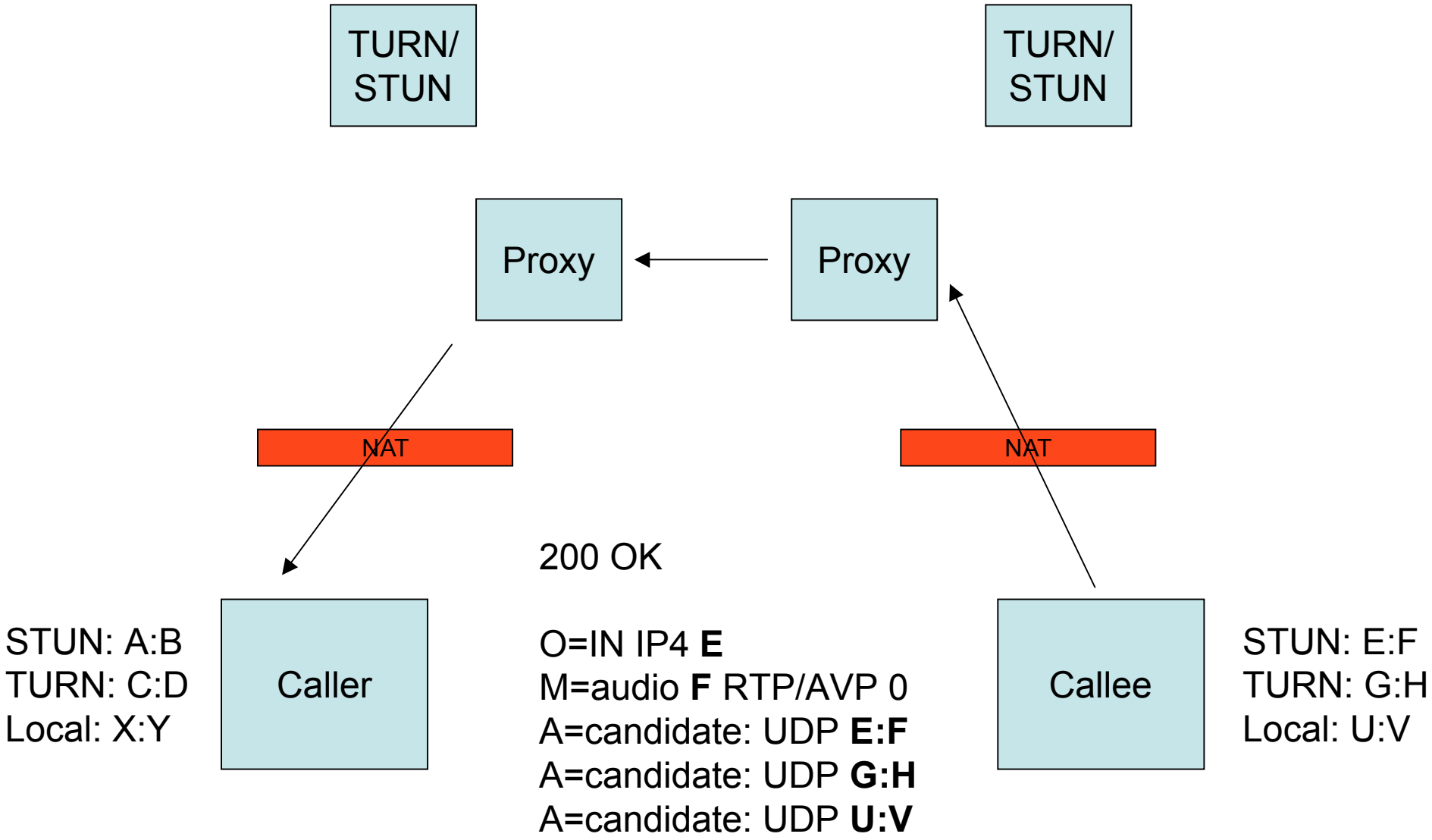

TURN/
STUN
TURN/
STUN
Proxy
Proxy
NAT
NAT
U:V
STUN: A:B
TURN: C:D
Local: X:Y
Caller
Callee gets STUN
And TURN addresses
From server
STUN: E:F
TURN: G:H
Callee

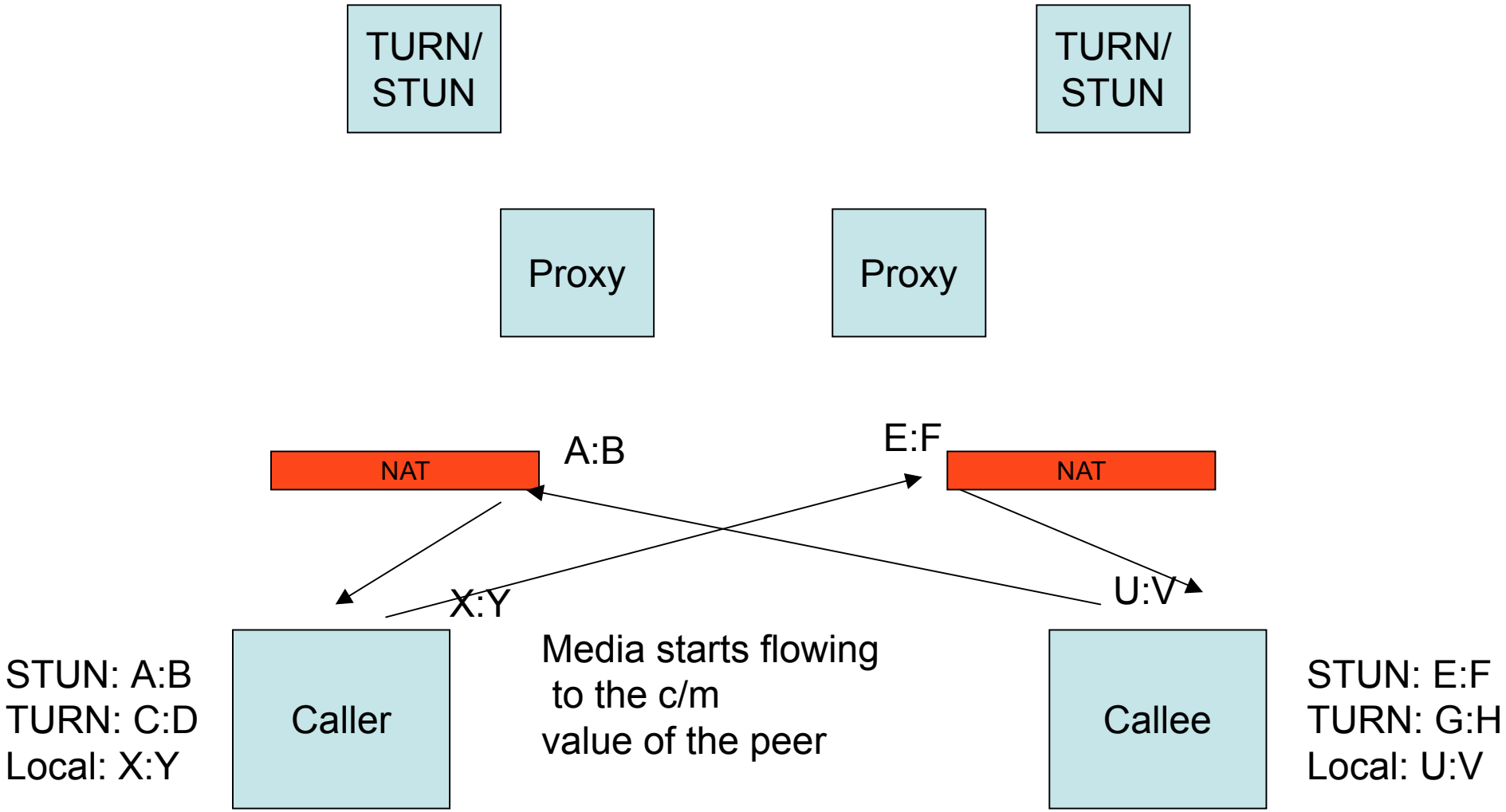

TURN/
STUN
TURN/
STUN
Proxy
Proxy
NAT
NAT
200 OK
STUN: A:B
TURN: C:D
Local: X:Y
Caller
O=IN IP4 G
M=audio H RTP/AVP 0
A=candidate: UDP E:F
A=candidate: UDP G:H
A=candidate: UDP U:V
Callee
STUN: E:F
TURN: G:H
Local: U:V


TURN/
STUN
C:D
G:H
TURN/
STUN
Proxy
Proxy
NAT
NAT
X:Y
U:V
STUN: A:B
TURN: C:D
Local: X:Y
Caller
Media starts flowing
immediately to the c/m
value of the peer
Callee
STUN: E:F
TURN: G:H
Local: U:V


TURN/
STUN
C:D
TURN/
STUN
Proxy
Proxy
A:B
NAT
NAT
X:Y
U:V
STUN: A:B
TURN: C:D
Local: X:Y
Caller
X:Y
Callee
STUN: E:F
TURN: G:H
Local: U:V
Connectivity checks
Ensue from callee to caller
STUN and TURN ones work
Same in reverse (not shown)


TURN/
STUN
TURN/
STUN
Proxy
Proxy
NAT
NAT
INVITE
STUN: A:B
TURN: C:D
Local: X:Y
Caller
O=IN IP4 **A**
M=audio **B** RTP/AVP 0
A=candidate: UDP **A:B**
A=candidate: UDP **C:D**
A=candidate: UDP **X:Y**
Callee


TURN/
STUN
TURN/
STUN
Proxy
Proxy
NAT
NAT
200 OK
STUN: A:B
TURN: C:D
Local: X:Y
Caller
O=IN IP4 **E**
M=audio **F** RTP/AVP 0
A=candidate: UDP **E:F**
A=candidate: UDP **G:H**
A=candidate: UDP **U:V**
Callee
STUN: E:F
TURN: G:H
Local: U:V


TURN/
STUN
TURN/
STUN
Proxy
Proxy
NAT
A:B
E:F
NAT
X:Y
U:V
STUN: A:B
TURN: C:D
Local: X:Y
Caller
Media starts flowing
to the c/m
value of the peer
Callee
STUN: E:F
TURN: G:H
Local: U:V

# How does this address each of the open issues?

- Issue #3: TCP
  - Candidate doesn't need RTP information since RTP never sent there!
  - When a TCP IP/port is listed as a candidate, you try to connect and check if it works
  - When it works, its "promoted" into the c/m lines and then you provide the SDP info as needed – just as if I did a re-invite updating a UDP to a TCP session w/o ICE
- Issue #4: what happened
  - What happened is always signaled in the m/c lines
- Issue #5: preconditions
  - No interactions anymore, since its normal precondition interactions with re-INVITEs
- Issue #6: middleboxes
  - Depends on what they do – QoS ones will work perfectly
  - Midcomish things still need to know candidates, or connectivity checks won't work, but call using default m/c will work

# Addressing the Issues

- Issue #7: dynamic RTP changes
  - Doesn't happen anymore unless signaled
- Issue #8: RTP demux
  - An implementation can avoid any demux by using separate value for c/m than candidates
  - Barrier sync through connectivity checks (later) so you never get STUN when you re-INVITE and move address into c/m
  - Drawback – no media while checks are running
- Issue #9: SRTP interaction
  - No longer an interaction – RTP never sent to candidates
  - Looks like a re-INVITE as far as SRTP is concerned

# FAQ

- Does this increase call setup delay?
  - No – INVITE/200/ACK as fast as previous versions
- Does this increase PDD?
  - No – media starts flowing to the c/m line as soon as INVITE is received (assuming it works)
    - If it doesn't work, PDD is suffered until a better candidate is found, followed by a re-invite to use it
    - This adds an RTT above existing mechanism for this case (corner case though)
- Does the re-INVITE always get sent?
  - No – only if the result of the checks produces a different address that you prefer

# Main technical question

- This approach depends on the caller figuring out when the callee's checks have succeeded
  - This requires a three-way handshake for the connectivity check
- Previously, RTP packet was used as the third leg of the check
  - And this introduced many of the problems
- Need a different approach
  - (1) Extend STUN with an ACK transaction of some sort (rfc3489bis)
  - (2) Use a second BindingRequest as the ACK
  - (3) Define a totally new protocol

# Question

- Does this seem like a reasonable change to pursue?
  - If so, will produce an update with the details