A Conferencing Architecture for 'Light-weight Sessions'

Van Jacobson
Steve McCanne
Sally Floyd

Lawrence Berkeley Laboratory

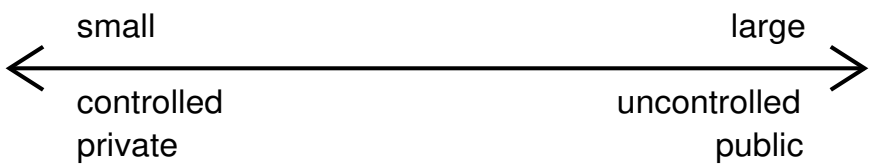
MICE seminar series
November 15, 1993

We've started to extend networking from machine–machine and human–machine communication to human–human communication.

There are many different styles of human–human communication. Evolution has shown them all to be necessary.

$\Longrightarrow$ We don't want a 'conferencing system', we want composable building blocks that can build many different styles of conferencing.

Decomposing the problem – one view:

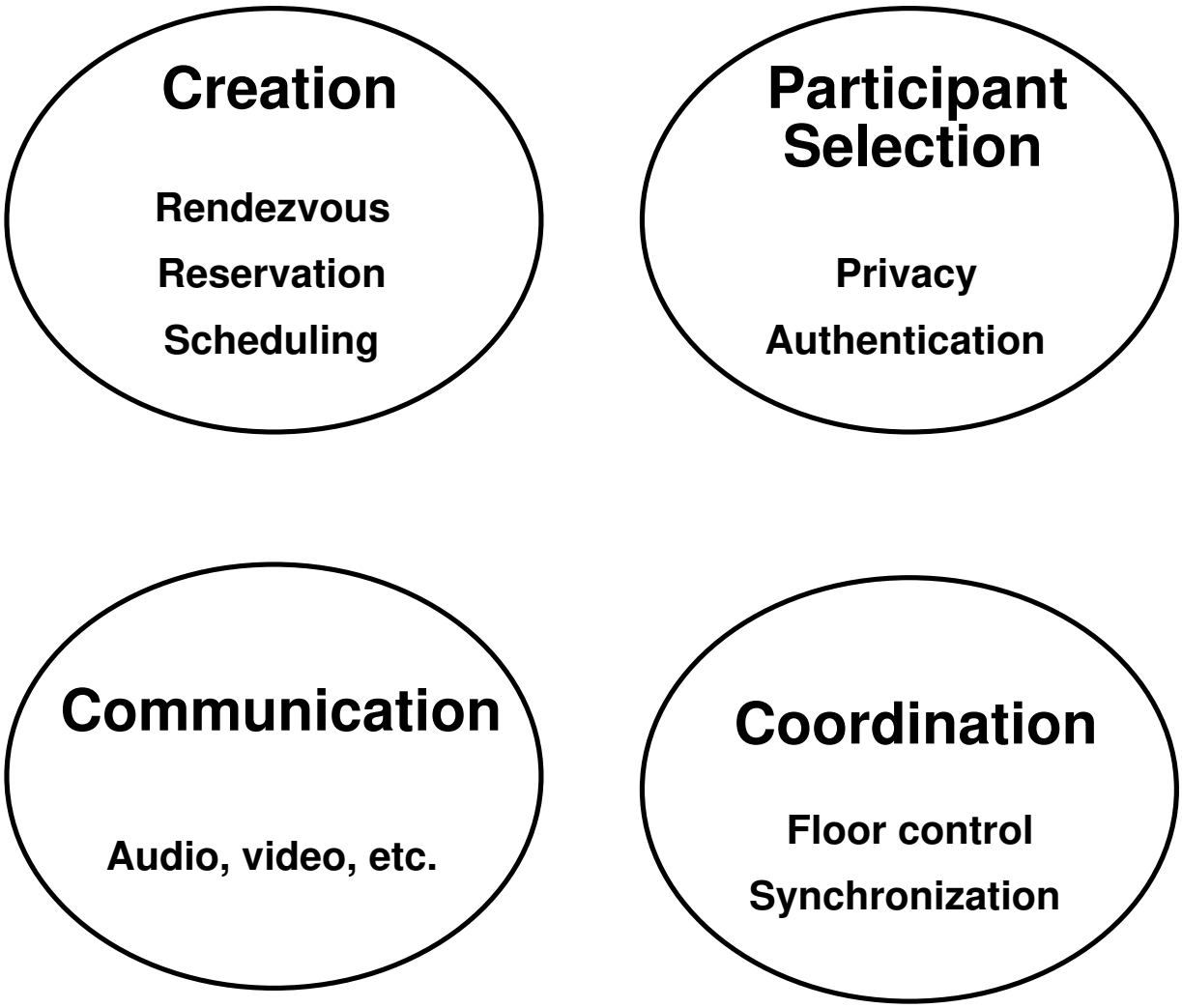


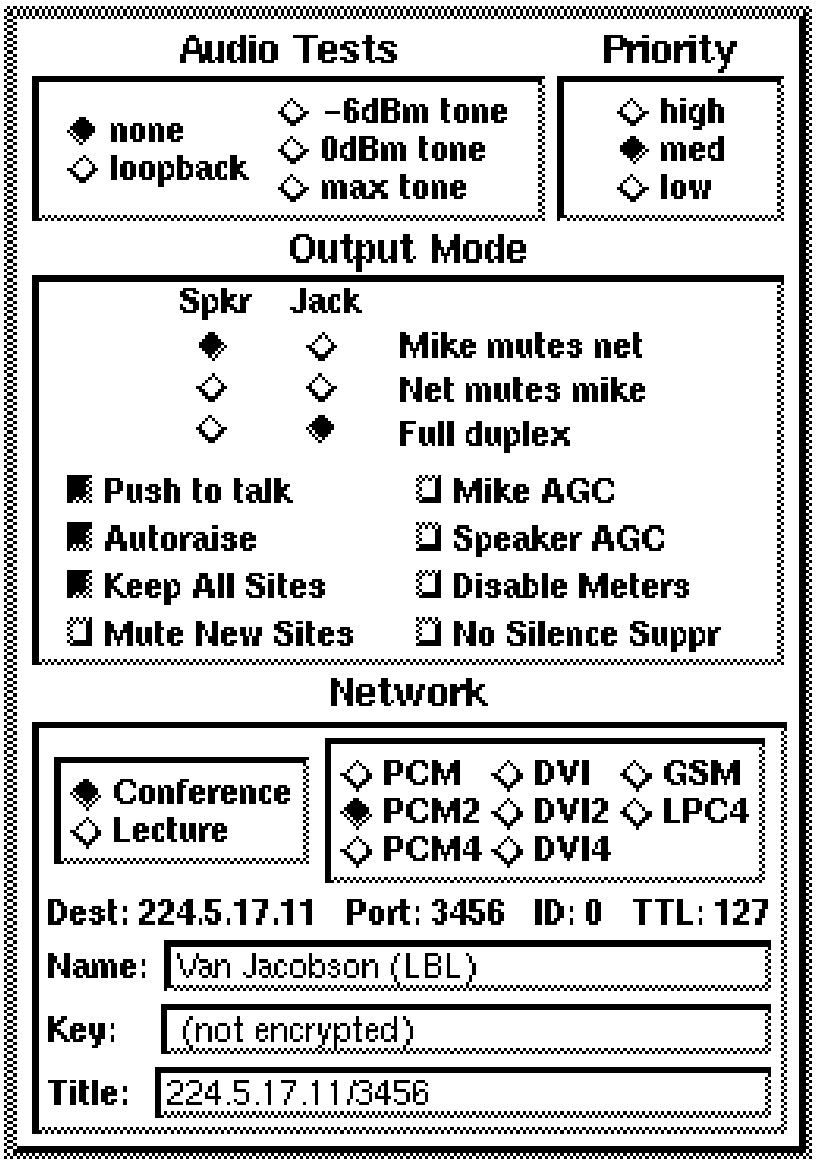
We think this decomposition is wrong:

- ‘Small’ and ‘large’ are too context dependent to be useful.

- Controlled/uncontrolled and private/public are completely orthogonal.

Decomposing the problem – our view:




Audio Tests
Priority
none
loopback
-6dBm tone
0dBm tone
max tone
high
med
low
Output Mode
Spkr
Jack
Mike mutes net
Net mutes mike
Full duplex
Push to talk
Autoraise
Keep All Sites
Mute New Sites
Mike AGC
Speaker AGC
Disable Meters
No Silence Suppr
Network
Conference
Lecture
PCM
DVI
GSM
PCM2
DVI2
LPC4
PCM4
DVI4
Dest: 224.5.17.11 Port: 3456 ID: 0 TTL: 127
Name:
Van Jacobson (LBL)
Key:
(not encrypted)
Title:
224.5.17.11/3456

Building blocks – participation

Encryption is the only way to generate privacy on an open Internet.

$\Longrightarrow$ Limiting/modifying participation is entirely a key distribution/revocation problem.

- This same problem appears in may other areas of the Internet architecture, not just conferencing (network management, resource reservation, online publication, etc.).

- The COCOM/ITAR regulations make it pointless to work on this problem in the US.

Privacy – what we've done

- Our tools all take an encryption key and can encrypt their data stream.

- Key can be set via typein, command line or internal message.

- Some thought was given to encryption in packet design:

  – If packets have seq.no., its starting value is random and it's the first thing in the packet.

  – If packets don't have seq.no., encrypted form of packet always starts with 64 bit random number.

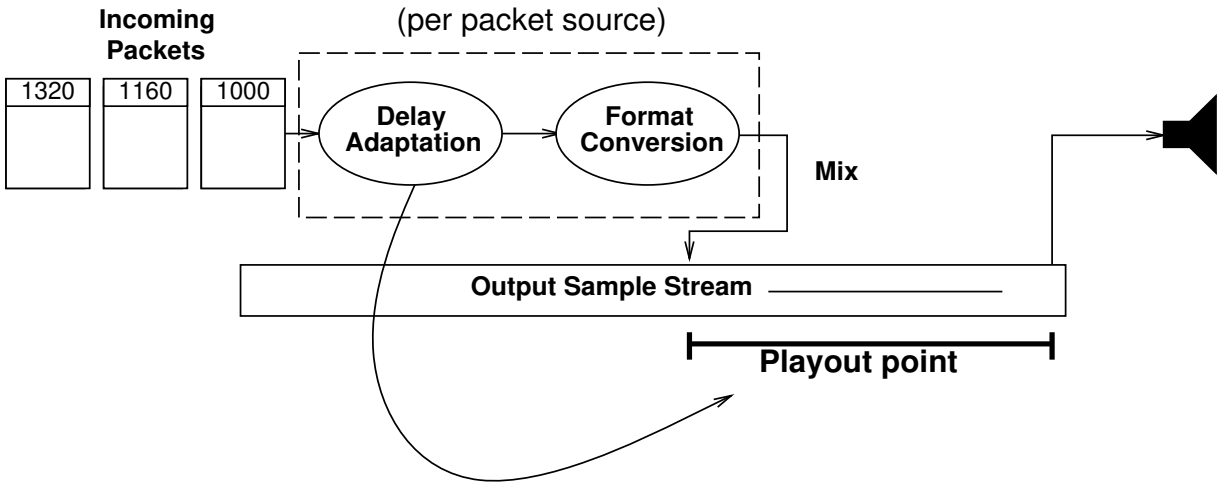
Privacy (cont.)

Sd has a (disabled) ‘lightweight security’ model:

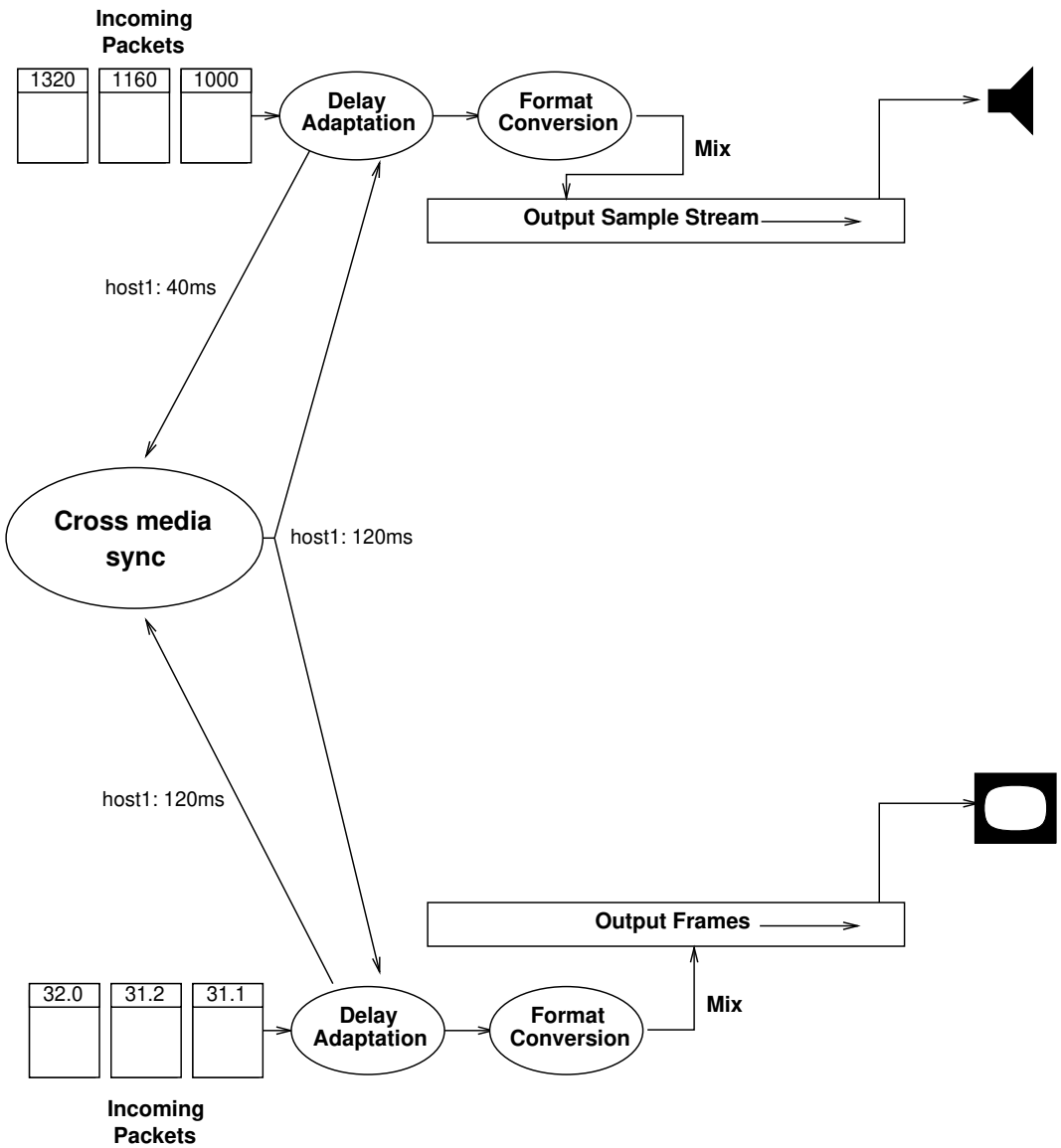
- Sd can be told to remember an arbitrary number of keys.
- Sd can be told to encrypt a session announcement with some key.
- If sd receives a packet it doesn’t understand, it tries to decrypt it with all the keys it knows. If one works, announcement is displayed and associated key is remembered. If session is opened, communication tools are started with that key.

Coordination – cross-media synchronization

Fundamental part of LWS model is that receivers adaptively remove the effect of network delays on sender's packets:



Cross-media synchronization (cont.)

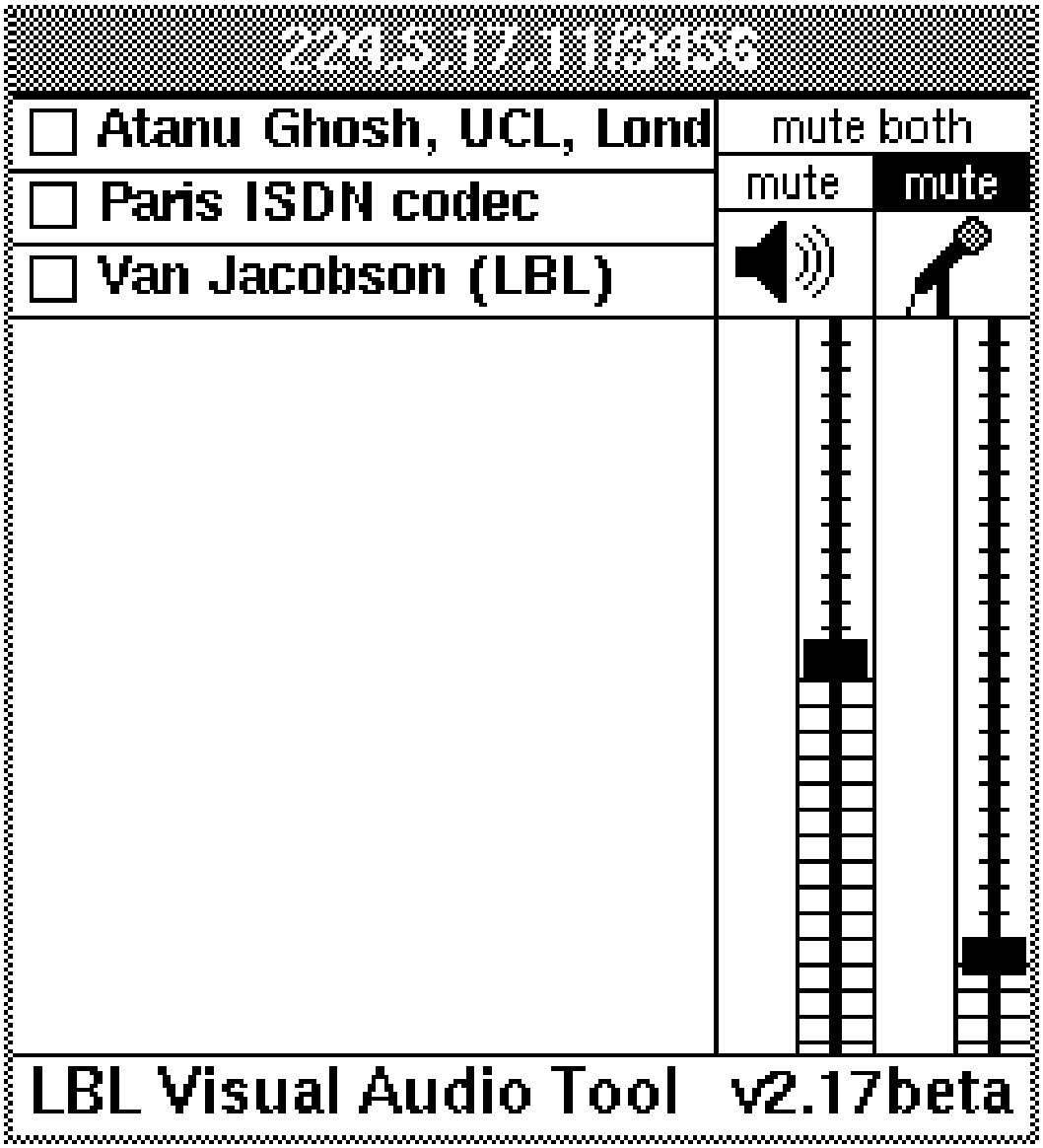


Cross-media synchronization (cont.)

Note that adaptation provides almost all the machinery needed for cross-media synchronization. Just need to add local agent that picks common playout point that both removes network effect and synchronizes media.

Note that playout-point changes must be infrequent (or will be audible / visible) so load on sync agent is light.

Also note that there's no reason to put clock time in data packets: low rate 'session' packets can contain media-to-clock time mapping. (There are also lots of reasons to use media rather than clock time in data packets.)


Atanu Ghosh, UCL, Lond
Paris ISDN codec
Van Jacobson (LBL)
mute both
mute
mute
LBL Visual Audio Tool v2.17beta

Coordination – floor control

Floor control is *very* context specific. E.g., a ‘class’, ‘moderated meeting’, ‘unmoderated meeting’ and ‘seminar’ all have different models of floor control. A class of ten year olds might use very different floor control from a class of PhD candidates.

It seems hard (to us) to specify floor control policy other than procedurally.

$\Longrightarrow$ Should be easy to code floor control agents and plug them in with other tools.

floor control (cont.)

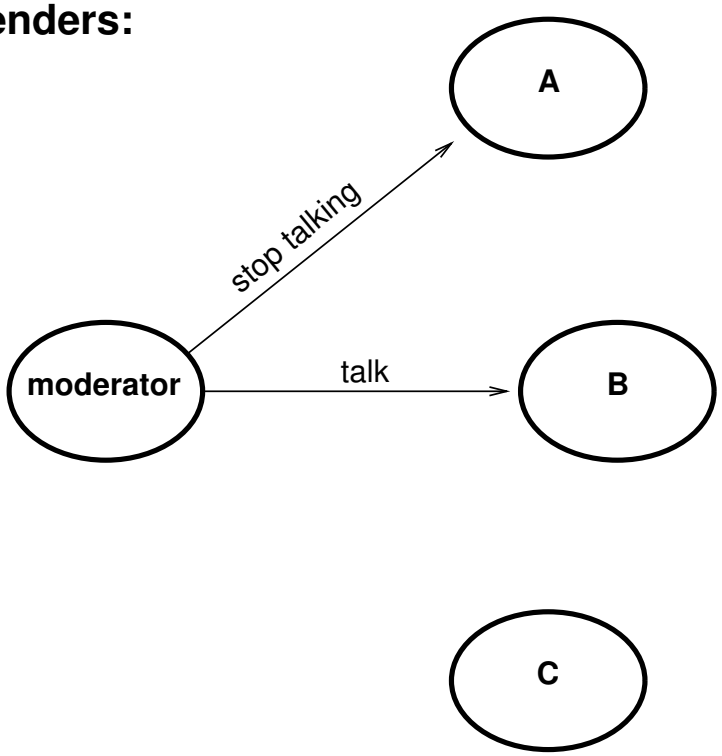
Enabling mechanism: sd announcement carries ‘session type’ attribute that causes appropriate floor control agent to be started together with media agents.

Also causes all agents to be started with port number of local ‘conference multicast socket’ so floor control agent can do its job.

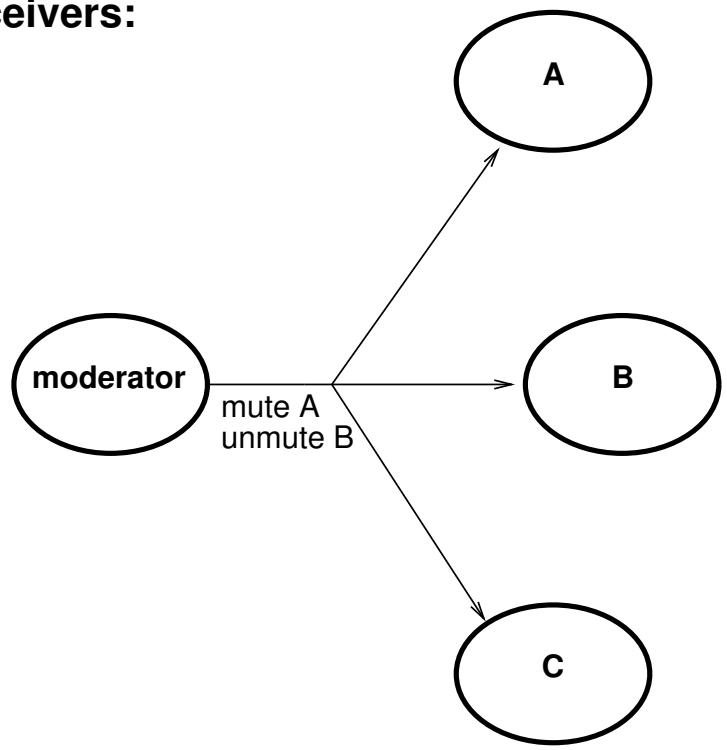
Aside: We’ve found need for two ‘local’ (reserved address at scope 0 through loopback interface) multicast sockets to compose tools: One is ‘global’ and coordinates access to shared hardware resources (mike, speaker, frame grabber, etc.). Others are per-conference and used to communicate floor control, sync info, key schedules, etc., among agents of the conference.

There are two models for doing floor control:

Control senders:



Advise receivers:

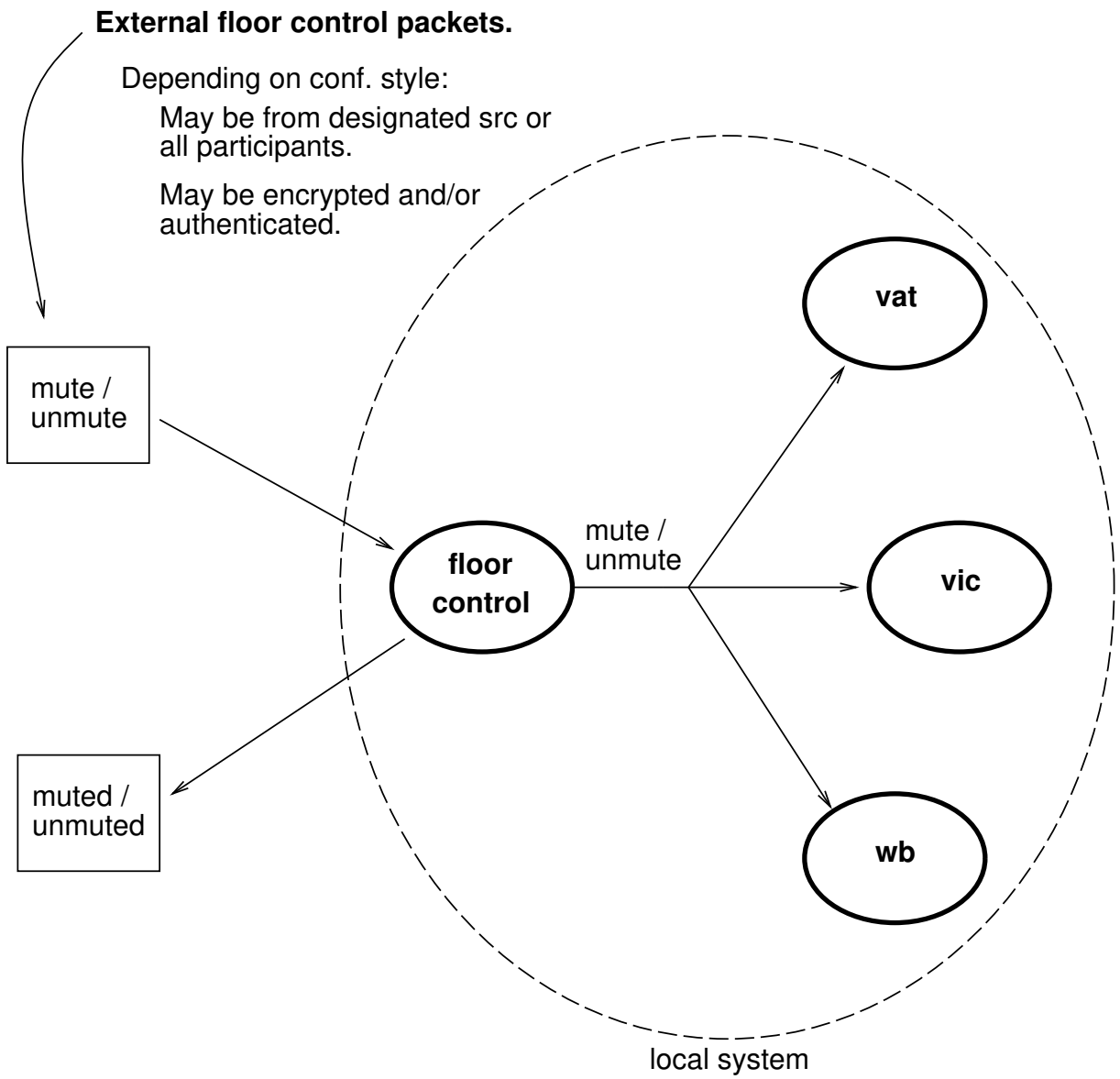


Implementing floor control (cont.)

If all participants agree, both models are equivalent.

If some participants don't play by rules, 'sender control' breaks badly. 'Receiver advice' still works (misbehaving senders just waste their own money by sending when no one is listening).

Our philosophy:

- Avoid 'agreement' – it's too heavyweight (people use delegation, rules of order, rough concensus, etc.).
- Use 'receiver advice' and avoid the fallacy of remote control.
- Build assistants, not masters.
- Leave participants in control of their privacy; try to minimize state information that has to be exported to network.

Our floor control architecture: