O2: Rethinking Open Sound Control

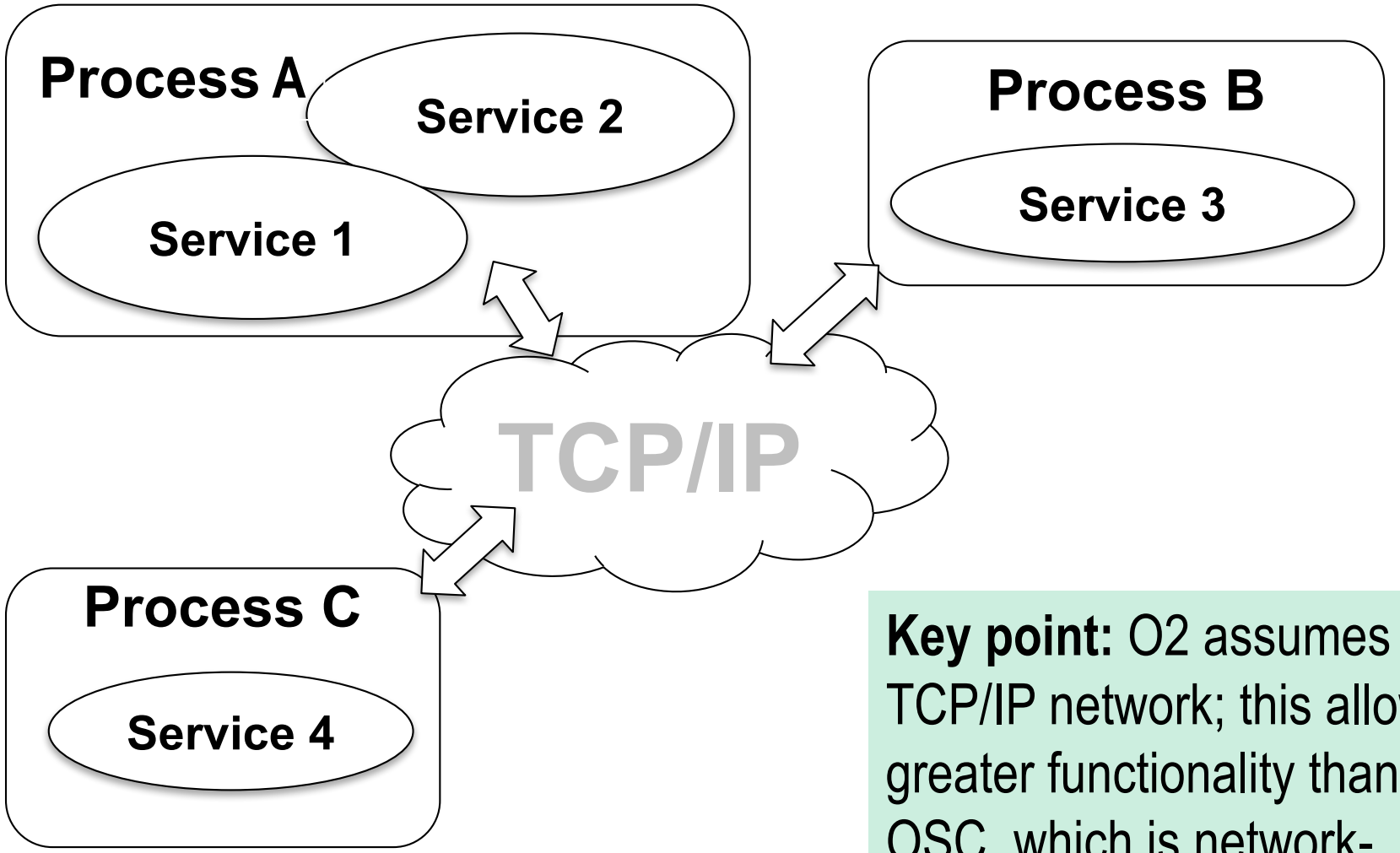
Roger B. Dannenberg
Carnegie Mellon University

Imagine…

- **Distributed real-time music/media applications that…**

- … address “sevices” by name, not numbers,
- … automatically find and connect themselves,
- … establish an accurate shared time base,
- … share low-latency, best-effort sensor data,
- … send guaranteed-delivery commands.

O2 System



Key point: O2 assumes TCP/IP network; this allows greater functionality than OSC, which is network-agnostic.

O2 Concepts

- **Host**: A computer attached to a local area network
- **Process**: A running program; there can multiple processes sharing a host
- **Application**: A collection of cooperating O2 *processes*. Applications should have unique *names,* allowing multiple applications to operate independently on a single network

O2 Concepts

- previous slide: **Host, Process, Application**
- **Service:** Processes can offer one or more *services*; each service in an application has a unique name and accepts *typed messages.*
- **Address:** an O2 address has the form:
  /service_name/aaa/bbb/ccc
- **Message:** an O2 address, timestamp, and list of typed parameters, e.g. we can write:

```
o2_send("/synth/noteon", 3.27,
        "iii", 1, 60, 100)
```

Putting It Together

- *o2_initialize*(“*application*”); *// one-time startup*
- *o2_add_service*(“*service*”); *// per-service startup*
- *o2_add_method*(“*address*”, “*types*”, *handler, data*);

- *o2_set_clock*(*clock_callback_fn*, *info_ptr*);

- *o2_send* (“*address*”, *time,* “*types*”, *val1*, *val2*, ...);
- *o2_send_cmd* (“*address*”, *time,* “*types*”, *val1*, ...);

Implementation

- Discovery:
  - All processes broadcast UDP “discovery” messages with IP address and port number
  - Receiver makes a TCP connection
  - Eventually, every process connects to every process
- Service Directory
  - Every process sends its service list to every discovered process (reliably over TCP).
  - Retransmit the list when it changes.

Implementation (2)

- Clock Synchronization:
  - Master provides a service: "`_cs`"
  - Others send their reply address to "`/_cs/get`" to get the master's time
  - Details:
    - subtract half the round-trip time,
    - pick best estimate,
    - smoothing,
    - clock rate estimation,
    - special cases for discontinuities

Implementation (3)

- Address patterns (like OSC):
  - `/service/??*/note[1-7]/{foo,bar}-[a-f]`
- We use a tree of hash tables for efficient lookup
- Special form to short-circuit pattern matching:
  - `!service/foo/note`
- Written as a library in C for portability, use by Max, Pd, Python, etc.
- Processes can use scheduled, time-stamped messages internally: no network overhead

Broadcast and Discovery

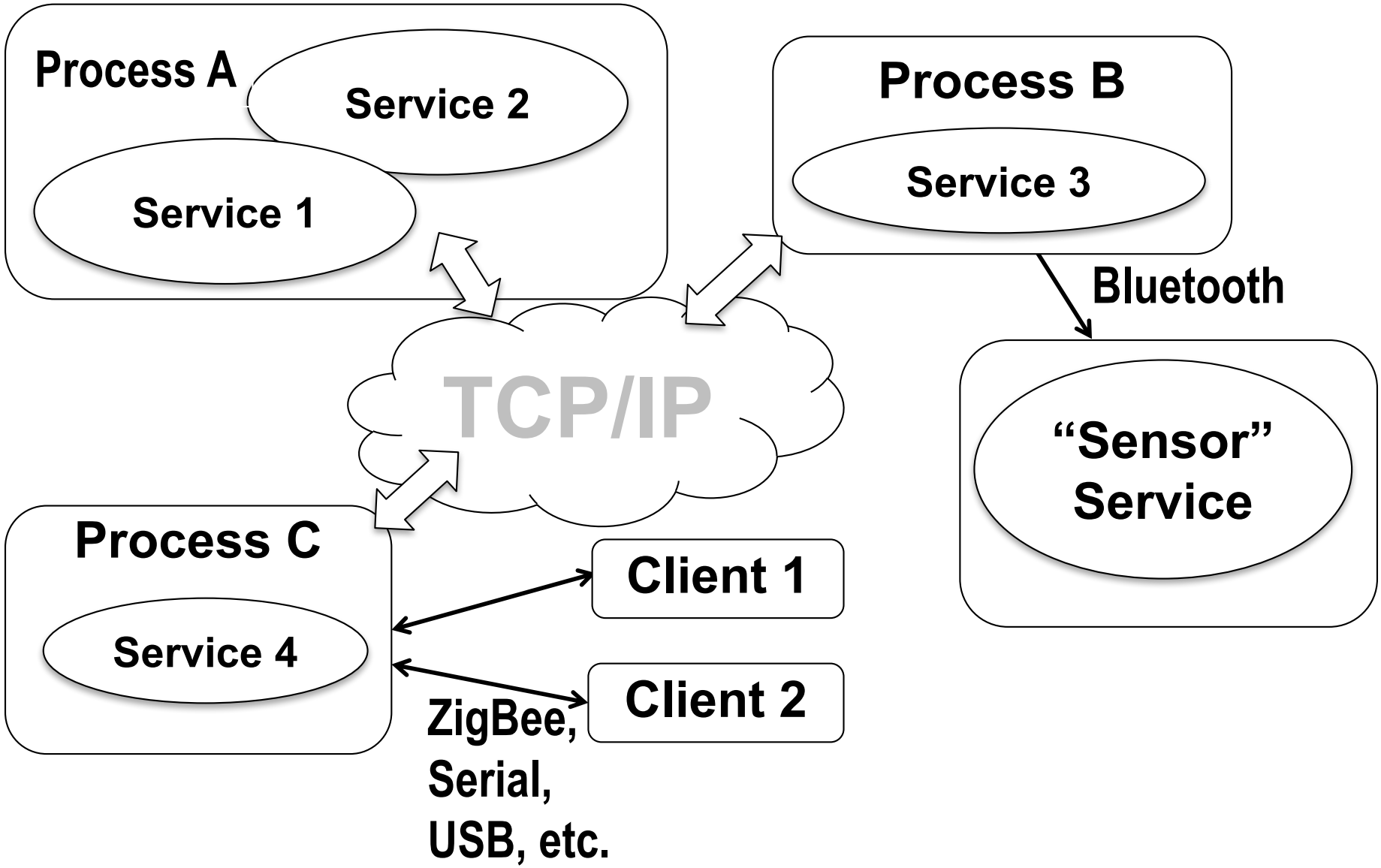
- Discovery in O2 is built on UDP broadcast messages.
- No broadcast => no discovery!
- We added a new feature “hubs”
  - If you identify an O2 process as your “hub” and provide its IP address and port number,
  - The “hub” will share all its discovery information
- So instead of broadcast messages, you can share the address of *one* process, and all processes will interconnect.
- Supports wide-area networking too.

Performance

- Dominated by network stack in the OS kernel
- Compared with liblo OSC implementation,
  - Extra time to process service names is negligible
  - We got about 77K msgs/sec on a single laptop: 2.4 GHz Intel Core i7
  - 13μs

What about Open Sound Control?

- You can receive and forward OSC messages from a particular O2 port to any named service
- You can forward O2 messages from a named O2 service to a particular OSC IP address and port



Example: CMU Laptop Orchestra

- See videos at:

2017: https://youtu.be/icLUJMM-11M

2018: https://youtu.be/L-Sar4D7lIY

Future Work

- Adapt to MAX, Pd, Python, JavaScript, etc.
- Provide “bridge” over Bluetooth, MIDI, ZigBee, etc., from O2 Process to embedded device.
- Multi-thread support to separate network operations from, say, real-time audio threads
- Work with Vesa Norilo on *audio* transport and audio (Kronos) server

https://github.com/rbdannenberg/o2

Conclusions

- O2 is a fast, flexible foundation for network and inter-process communication in music and media applications.
- Solves several problems of OSC:
  - No more manually typing in dynamic IP addresses to configure systems,
  - No risk of dropped commands (“start”, “note-off”),
  - Accurately timed message delivery – at last.