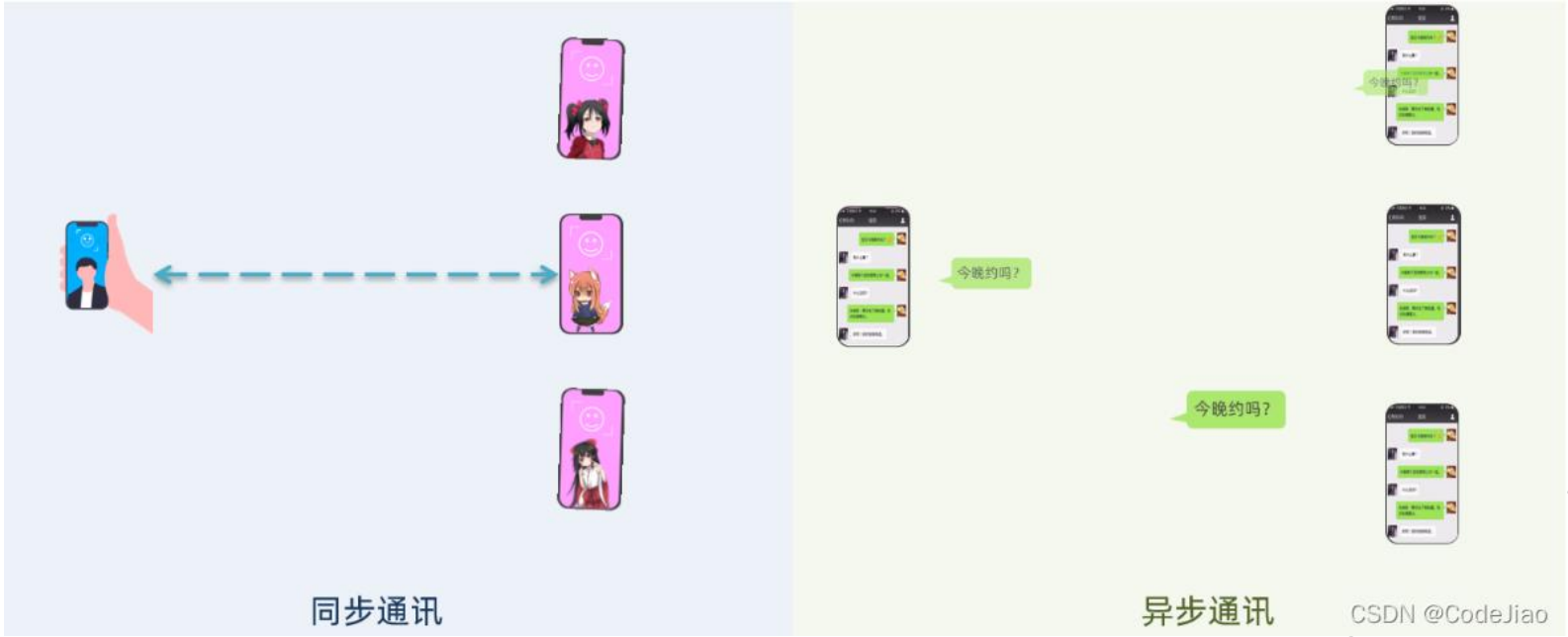
Background

- 同步和异步
  - 同步通信：在同步通信模式下，发送方和接收方之间的通信是阻塞的，也就是说发送方会等待接收方完成某个操作后才能继续执行。在同步通信中，发送方发送消息后会一直等待接收方确认接收并完成相应的处理，只有在这个过程完成后，发送方才能继续执行后续操作。
  - 异步通信：在异步通信模式下，发送方和接收方之间的通信是非阻塞的，也就是说发送方发送消息后可以立即继续执行后续操作，而不需要等待接收方的响应。在异步通信中，发送方和接收方可以并发执行，彼此之间不会阻塞。



IPC

- 在 seL4 中， IPC是传输少量数据的同步机制。
- 当Send时，如果接收端没有调用Recv，那么当前线程会被阻塞。
- 当Recv时，如果发送端没有调用Send，那么当前线程会被阻塞。
- seL4中的IPC通过内核对象 endpoint 来实现，可以类比网络通信中的端口概念。用户态任务可以通过 endpoint 的 cap 来发送 IPC 消息。
- endpoint 由一个等待发送或等待接收消息的线程队列组成，可以参考下面这个例子：
  - 在一个 endpoint 上有 n 个线程在等待接收消息，如果有 n 个线程在端点上发送了 n 个消息，则之前等待的 n 个线程将接收到消息并被唤醒，如果第 n + 1 个 发送者再次发送，则会被阻塞。

IPC Syscall

- seL4_Send：发送消息，会阻塞直到消息被接收消费掉。
- seL4_NBSend：发送消息，只有在已经有接收者阻塞在队列中时才会发送成功，否则发送失败，且不会返回发送结果，不会阻塞。
- seL4_Recv：接收消息，会阻塞直到接收到消息。
- seL4_NBRecv：接收消息，不会阻塞。
- seL4_Call：相当于将 seL4_Send 和 seL4_Recv 组合。只有一个区别：在接收阶段，线程被阻塞在一个单独的叫做 reply cap 上，而不是 endpoint cap。
- seL4_Reply：通过存储在接收线程 TCB 中的 reply cap 调用，会发送一个 IPC 消息给客户端并将客户端唤醒。。
- seL4_ReplyRecv：与上面的类似，只不过回复之后会重新阻塞接收。

- 由于 TCB 只有一个空间来存储一个 reply cap，因此如果服务器要为多个客户端提供服务的话，需要调用 seL4_CNode_SaveCaller 来将 reply cap 存到一个空的 slot 中。

相关数据结构

- IPC Buffer：每个线程都有一个缓冲区，包含了IPC消息的有效载荷，由数据和 cap 组成。发送方指定消息长度，内核在发送方和接收方的 IPC Buffer 之间复制数据。
- Data transfer：IPC Buffer 包含了一个有界区域的消息寄存器（MR）用于传输数据，每个寄存器长度为机器字长，最大的消息大小在 libsel4 中被定义为 seL4_MsgMaxLength。
  - 用户态还可以通过 seL4_SetMR 和 seL4_GetMR 来设置或获取消息。小的消息可以直接通过寄存器来发送而不用进行复制操作。
- Message Info：seL4使用数据结构 seL4_MessageInfo_t 来描述被编码后的 IPC 消息。包含了以下几个字段。
  - length ：消息中的 MR 数量。
  - extraCaps ：消息中包含的 cap 的数量。
  - capUnwrapped ：标记内核 unwrap 的 cap。
  - label：传输的有效载荷。
- Cap transfer：IPC还可以在传输数据的同时将 cap 也进行传输，这被叫做 cap transfer。
- Badges：接收端如何区分消息是从哪个发送端发送过来的，这时候可以使用Badges标记发送端，具体标记值由自己定，当带这个标记的发送端发送消息时， 标记被传递到接收端的标记寄存器中，此时可以在接收端检查标记寄存器，判断发送端。

seL4: Notification

廖东海

ctrlz.donghai@gmail.com

Notification

- 通知机制允许进程向其他进程发送异步信号，主要用于中断处理和同步访问共享缓冲区。
- Notification objects：Notification objects 是一个内核对象，用户态程序通过指向对象的 cap 来发送和接收信号。该对象由一个数据字（看作一个二值信号的数组）和一个等待通知的TCB队列组成。这个对象有三种状态：
  - 等待：有TCB等待在此对象上的信号。
  - 激活：TCB已发出有关此通知的数据。
  - 空闲：没有TCB排队，且从上次置为空闲态后没有TCB发送通知数据。

User Interface

- Signalling：当一个任务在一个通知对象上发送信号时（使用 seL4_Signal），表现出来的行为跟通知对象的状态有关：
  - 等待：TCB等待队列的头部被唤醒，badge被发送到该TCB上，如果队列为空，则对象转化为空闲状态。
  - 激活：用于向通知对象发送消息的功能标记与通知数据字按位与。
  - 空闲：数据字设置为用于发送信号的能力标志，对象转换为激活状态。
- Waiting：通过 seL4_Wait 等待一个通知，表现出来的行为跟通知对象的状态有关：
  - 等待：TCB入队尾进行等待。
  - 激活：TCB接收到数据字，数据被置为0，状态转换为空闲状态
  - 空闲：TCB入队等待，并且转换为等待状态。
- Polling：任务可以调用 seL4_Polling 来轮询队列，效果相当于非阻塞的 seL4_Wait，会立即返回结果。

Thanks for Listening.