

System	OS / Language	Concurrent	Inter-Thread Sync. Sends	Reentrancy	Non-Blocking Comms.	EQ
E	Language	No	N/A	Recursive	Send + Recv	Yes
Cajita <i>et al.</i> ²	Language	No	N/A	Recursive	Send + Recv ³	Yes
Joe-E	Language	No	N/A	Recursive	Send + Recv ³	Yes
Emily	Language	No	N/A	Recursive	No	Yes
Tamed Pict	Language	Yes	No	R.+Concurrent	Send + Recv	No
EROS	OS	Yes	Yes	No	No	Yes
CapROS	OS	Yes	Yes	No	No	Yes
KeyKOS	OS	Yes	Yes	No	No	Yes
Coyotos	OS	Yes	Yes	No	Send ⁴	Yes
seL4	OS	Yes	Yes	No	Send ⁴	No
Plash	(Virt.) OS	Yes	Yes	R.+Concurrent ⁵	Send + Recv	Yes
Annex (2008-09)	(Dist.) OS	Yes	Yes	R.+Concurrent ⁵	Send	No

Figure 1: A Taxonomy of Current Object-Capability Systems¹

¹Thanks to Bill Frantz, David-Sarah Hopwood, Matej Kosik, Charles Landau, Alex Murray, Mark Seaborn and David Wagner who helped complete this table.

²Cajita *et al.* are JavaScript subsets. In some JavaScript implementations, use of functions like `window.setTimeout` in conjunction with functions like `alert` can produce effects akin to the interleaved execution of multiple threads running concurrently [?]. We restrict our attention to circumstances in which this corner-case does not arise.

³The `ref_send` library provides non-blocking communication in Joe-E and Cajita

⁴This facility is provided in seL4 and Coyotos through their “non blocking” Send operations which are best-effort sending primitives that don’t notify the sender if sending fails for some reason [?, ?]. seL4 also includes an “asynchronous” Send operation; however, this cannot be used to transfer capabilities.

⁵Plash and Annex have similar reentrancy. In both, a single thread is allocated at minimum to each object. However, the thread of an object

- A single object-capability system relies on a single shared TCB. Hence, We restrict our attention to the non-distributed parts of each system above.
- A system is *concurrent* if it comprises more than a single thread of control.
- A system that has *inter-thread synchronous sends* if it is concurrent and allows one object to synchronously send a message to another that doesn't share the same thread of control. Such sending will block the sender until the recipient is ready for the message.
- A system may exhibit the following forms of *reentrancy*: *concurrent*, *recursive*, neither or both. A system exhibits concurrent reentrancy if it is concurrent and a single object, unless going to special lengths to prevent it, can be executed by multiple threads simultaneously. A system exhibits recursive reentrancy if a single object, unless going to special lengths to prevent it, can be executed recursively by a single thread.
- A system may provide *non-blocking communications* for *sending*, *receiving* or both. It is not required for a sender to be notified when a non-blocking send completes and in many systems no such notification is provided.
- A system provides the equality primitive *EQ* if it provides a facility to determine whether two arbitrary capabilities refer to exactly the same object, even if this facility is not universally available to every object in the system.

∞

that blocks on a synchronous receive (in Plash) or synchronous send or receive (in Annex) is free to process other incoming messages thereby creating a situation in which it may service multiple requests at one time either from concurrent or recursive invocation.