	TCP		UDP		IPX	
	Client	**Server**	**Client**	**Server**	**Client**	**Server**
socket	+	+	+	+	+	+
bind		+		+		+
listen		+				
set_non_blocking		[+]				
connect	+		[+]		[+]	
shutdown	[+]	[+]				
close	+	+	+	+	+	+

socket	int sockfd = socket (
	  PF_INET
	, SOCK_STREAM
	, IPPROTO_TCP);
		@see /TCP/Client/socket	int sockfd = socket (
			  PF_INET
			, SOCK_DGRAM
			, IPPROTO_UDP);
				@see /UDP/Client/socket	int sockfd = socket (
					  PF_IPX
					, SOCK_DGRAM
					, 0);
						@see /IPX/Client/socket

bind		sockaddr_in saddr;
		uint16_t port = /* port */;
		uint32_t addr = /* 32-bit IP4 address */;
		
		memset(& saddr, 0, sizeof(saddr));
		saddr.sin_family = PF_INET;
		saddr.sin_port = htons(port);
		saddr.sin_addr.s_addr = htonl(addr);
		
		int rc = bind(sockfd
		  , & saddr
		  , sizeof(saddr));
				@see /TCP/Server/bind		

listen		int backlog = /* SOMAXCONN or less */; rc = listen(sockfd, backlog);				
set_non_blocking		int flags = fcntl(_fd, F_GETFL, 0); fcntl(_fd, F_SETFL, flags | O_NONBLOCK) >= 0;				

connect	sockaddr_in server_addr;
	uint16_t port = /* port */;
	uint32_t addr = /* 32-bit IP4 address */;
	
	memset(& server_addr, 0, sizeof(server_addr));
	stSockAddr.sin_family = PF_INET;
	server_addr.sin_port = htons(port);
	server_addr.sin_addr.s_addr = htonl(addr);
	
	int rc = connect(sockfd
	  , & server_addr
	  , sizeof(server_addr);					

shutdown	shutdown(sockfd, SHUT_RDWR);	shutdown(sockfd, SHUT_RDWR);				
close	close(sockfd);	close(sockfd);	close(sockfd);	close(sockfd);	close(sockfd);	close(sockfd);

Application loop	while(1) {
	  send
	  recv
	}
		while(1) {
		  accept
		  recv
		  send
		}
			while(1) {
			  sendto
			  recvfrom
			}
				while(1) {
				  recvfrom
				  sendto
				}
					while(1) {
					  sendto
					  recvfrom
					}
						while(1) {
						  recvfrom
						  sendto
						}