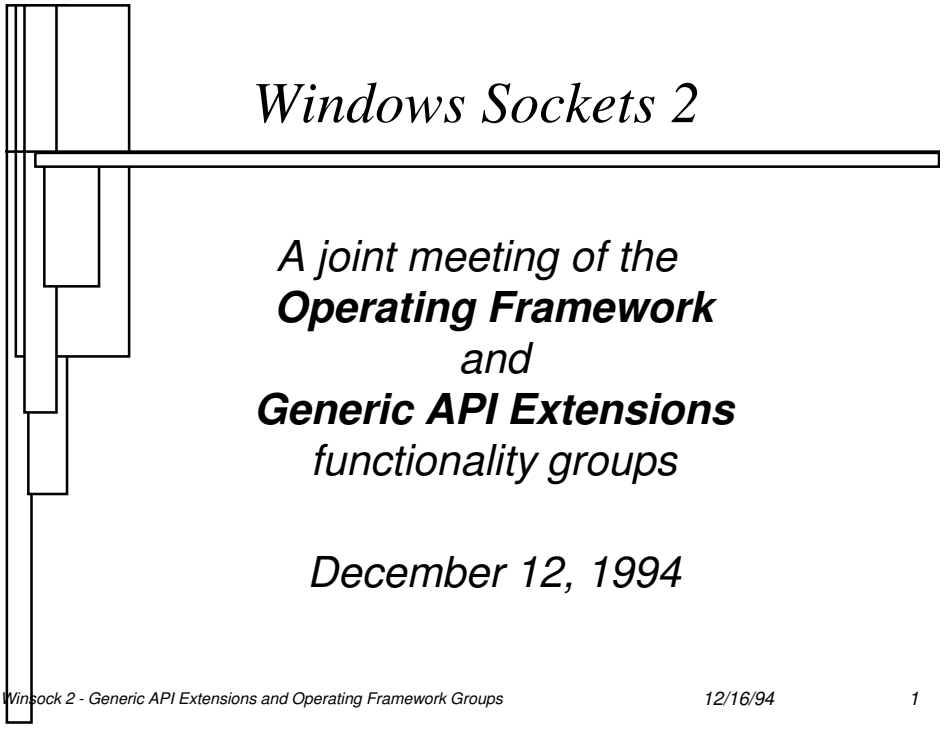
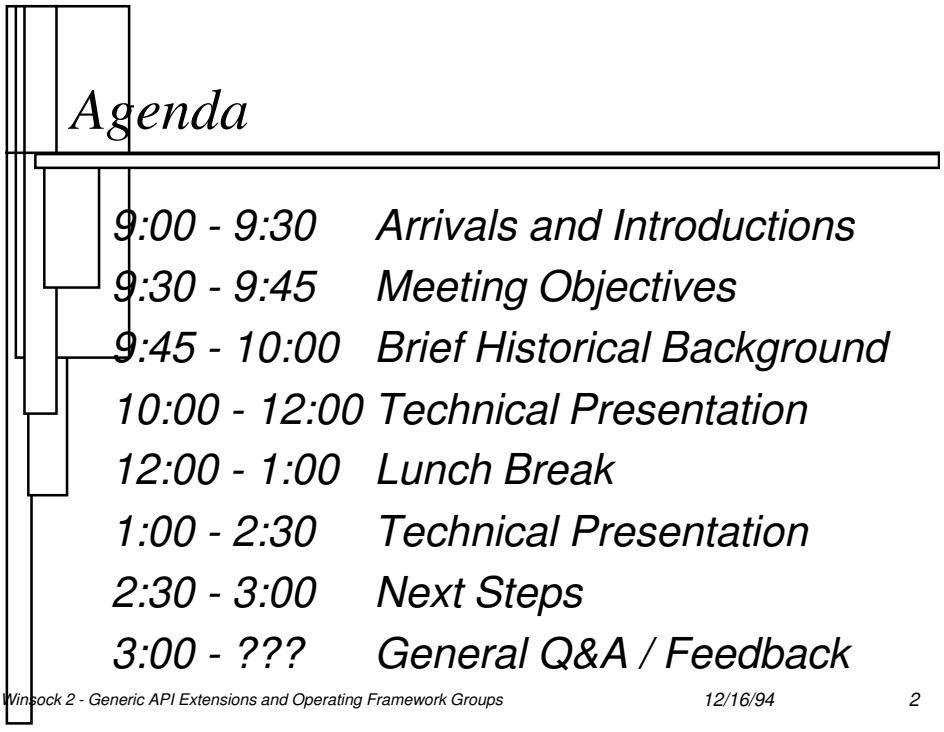


Windows Sockets 2

A joint meeting of the
Operating Framework
and
Generic API Extensions
functionality groups

December 12, 1994



Agenda

9:00 - 9:30	*Arrivals and Introductions*
9:30 - 9:45	*Meeting Objectives*
9:45 - 10:00	*Brief Historical Background*
10:00 - 12:00	*Technical Presentation*
12:00 - 1:00	*Lunch Break*
1:00 - 2:30	*Technical Presentation*
2:30 - 3:00	*Next Steps*
3:00 - ???	*General Q&A / Feedback*

Technical Presentation Outline

- *Requirements Summary*
- *New Features by Category*
  - *Simultaneous access to multiple transport providers*
  - *Latency and throughput enhancements*
  - *Quality of Service*
  - *Generic functionality extensions*
- *Open Issues*

Meeting Objectives

- *Report on a joint effort to produce core elements of a strawman spec*
- *Answer questions and solicit feedback*
  - *Goal: API and SPI specs published by Christmas*
- *Refine the process for moving Winsock 2 forward*

A Brief Historical Perspective

- How the strawman team came to be
- Motivations
- Work methods and time span

Technical Presentation

- Requirements Review

New Features by Category

- Simultaneous access to multiple transport providers
- Latency and throughput enhancements
- Quality of Service
- Generic functionality extensions

Open Issues

Operating Framework Requirements

Requirement	Status
1) Operating System versions - Windows 3.1, Windows 3.11, Windows 95, Windows NT	√
2) Architecture	√
3) Multiple protocol stacks on a single host via a single DLL	√
4) Mechanism for determining version of a specific Window Sockets DLL	Partial
5) Configuration	√
6) Clearinghouse for address family, protocol, socket type, and socket option ordinals	√

Generic API Requirements

Requirement	Status
1) C++ class libraries	Deferred
2) Callbacks	√
3) Shared sockets	√
4) More socket types	√
5) Send and recv from preemptive context	√
6) Socket groups	√
7) User data exchange during conn setup	√
8) Apps can establish host name/addr	Disagree
9) Quality of service	√
10) Conditional acceptance	√
11) Allow optional winsock extensions	Partial

Generic API Requirements (cont)

Requirement	Status
12) Prevent reentrant msgs while blocking	blocking hook
13) Define std error text	pending
14) Define std debugging paradigm	needed?
15) Way to get DLL version	√
16) Asymmetric send / recv msg size	pending
17) Scatter / gather [readv(), writev()]	needed?
18) Asynchronous (overlapped) I/O	√
19) Sock opts for send / recv timeouts	event objects
20) Four byte per socket context value	disagree
21) Support for msg-oriented (partial) recv	√
22)Re-enabling after error in FD_READ	needed?

Technical Presentation

- Requirements Summary
- ➢ New Features by Category
  - ⇒Simultaneous access to multiple transport providers
  - • Latency and throughput enhancements
  - • Quality of Service
  - • Generic functionality extensions
- Open Issues

Simultaneous access to multiple transport providers

- *SP discovery and usage model*
- *Standardized Winsock2 DLLs*
- *Service Provider Interfaces (16 and 32 bit)*

Multiple Simultaneous Protocols: A Paradigm Shift

- *Old: well-known socket types and protocol values*
  - *Apps hard coded to certain protos because of their well-known attributes*
- *New: don't care about socket type and protocol value*
  - *Apps select protos on the basis of their attributes - need not be hard coded!*

Multiple Protocols from an Application's Perspective

- *Client / Server apps*
  - *Servers listen on all suitable protocols*
  - *Clients connect using any suitable protocol*
  - *Address discovery via WS2 name resolution*
- *Peer to peer apps*
  - *Address book contains typed addresses*
    - *Example: phone #, TCP/IP addr, ATM addr*
  - *Use suitable protocols that match address type (i.e. address family)*

Choosing One or More Protocols

- ***WSAEnumProtocols()***
  - *Input: filter (optional), buffer, buffer length*
  - *Output: filled in buffer, revised buffer length*
- *PROTOCOL_INFO structs*
  - *One per protocol (usually)*
  - *Describes protocol attributes and behavior*
  - *Supplies all params needed for **socket()** or **WSASocket()***

Protocol Attributes

Service Flags

Connectionless	*Reliable*	*Guaranteed order*
Message oriented	*Pseudo stream*	*Graceful close*
Expedited data	*Connect data*	*Broadcast capable*
Multicast capable	*QOS supported*	*Encryption supported*
Interrupt capable	*Unidirectional send*	*Unidirectional recv*

Lots of space reserved for future expansion

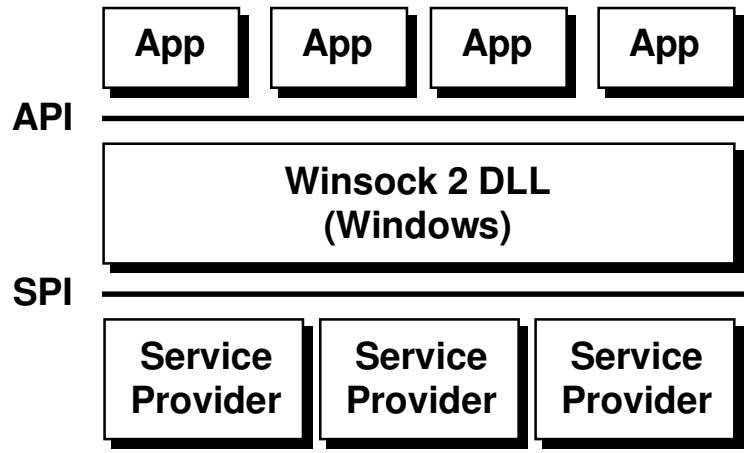
Protocol Attributes (cont.)

Parameter Values

Address family	*Address length*	*Socket type*
Protocol value	*Protocol name*	*Protocol version*
Provider ID	*Message Size*	*Name space IDs*

- *Multi-behaviored Protocols*
  - *One PROTOCOL_INFO entry per behavior*
  - *All entries have same Protocol Value*
  - *Socket type used to distinguish*

Winsock 2 Architecture



SPI Concepts (WS2SPI.DOC, section 1)

- *One winsock2 DLL per platform*
- *Thin winsock2 DLL routes to providers*
- *Architectural freedom for providers*
- *Provider IDs*
- *Provider handles any necessary buffering*

32-Bit vs. 16-Bit SPI (WS2SPI.DOC, section 2.7)

- Differences minimized; portability maximized
- Blocking models differ
- Handle management differs

Installation APIs (WS2SPI.DOC, section 5)

- WPUInstallProvider()
- WPUDeinstallProvider()
- Do not install files; only inform winsock2 DLL.
- Abstract storage of config info

Helper Upcalls (WS2SPI.DOC, section 4)

- *Thread Management*
  - *Thread ID*
  - *Queue APC/callback*
- *Handle Management*
  - *creation of unique socket handles*

Technical Presentation

Requirements Summary

- *New Features by Category*
  - *Simultaneous access to multiple transport providers*

  ⇒*Latency and throughput enhancements*
  - *Quality of Service*
  - *Generic functionality extensions*

Open Issues

Latency and throughput enhancements

- *Overlapped socket operations*
- *Send and recv from within preemptive contexts*
- *More efficient notification: events, APCs, callbacks*

Overlapped I/O

- *Fewer data buffer copies*
- *Multiple simultaneous pending I/O requests per socket*
- *Thread-safe callback mechanism in preemptive environments*
- *Flexible notification options (APCs or event objects)*

Notification Mechanisms

- *select()*
- *WSAAsyncSelect()*
- *WSAEventSelect()*
- *APCs and event objects (overlapped i/o completion)*

Technical Presentation

Requirements Summary

- *New Features by Category*
  - *Simultaneous access to multiple transport providers*
  - *Latency and throughput enhancements*

  ⇒*Quality of Service*
  - *Generic functionality extensions*

Open Issues

Quality of Service

- Flow specs
- Socket groups
- Socket prioritization

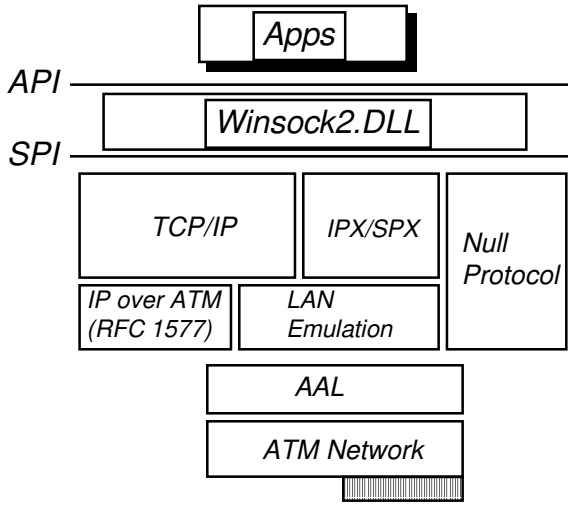
Flow Spec

- Usage Model
  - characteristics of a uni-directional flow
  - apps request and networks respond
  - map QOS Name to Flow Spec
  - coincident on connection set-up time
  - query established Flow Spec
  - get notified for any changes

Flow Spec (continued)

- Categories of Flow Spec parameters
  - Bandwidth (peak, average, burst length)
  - Latency (end-to-end delay, delay variation)
  - Level of service guarantee
  - Cost
  - Provider-specific parameter

Example: Winsock 2.0 over ATM



Socket Groups

- Establish relationship for a particular set of sockets
- Socket group created/joined at connection set-up time
- Group QOS maps to the underlying call
- Relative priorities among sockets within a group

Technical Presentation

Requirements Summary

- New Features by Category
  - Simultaneous access to multiple transport providers
  - Latency and throughput enhancements
  - Quality of Service

  ⇒Generic functionality extensions

Open Issues

Generic functionality extensions

- *Shared sockets*
- *Connection establishment*
  - *Conditional Acceptance*
  - *Exchange of user to user data*
- *Bi-directional flags parameter in receive functions*

Shared Sockets

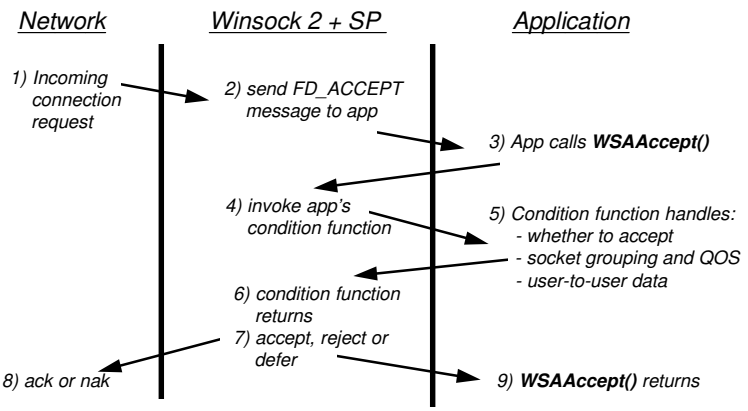
- *Single socket with multiple handles*
  - *Use **WSADuplicateSocket()** to create*
  - *Parameters:*
    - *source socket handle*
    - *destination task handle*
  - *Returns: socket handle valid only in context of destination task*
  - *IPC needed to get destination task handle and to supply new socket handle*

Shared Sockets (cont)

- I/O must be coordinated by apps
- Notification is ***independent*** to each handle
- Socket options and ioctl's are common

Conditional Acceptance

- ***WSAAccept()*** condition function



Exchanging Connect Data

- *Support for user data exchange during connect sequence*
  - *Initiator uses **WSAConnect()***
    - *supplies buffers for inbound and outbound data*
    - *inbound buffer only valid after FD_CONNECT*
  - *Listener uses **WSAAccept()***
    - *condition function used to get/supply buffers*
  - *Only works if supported by protocol*

Making recv flags IN OUT

- *Use of 'MORE' flag by some protocols*
- *Winsock 1.1 **recv()** and **recvfrom()** have flags parameter as INPUT only*
- ***WSARecv()** and **WSARecvFrom()***
  - *flags parameter is IN OUT*
  - *useable by both overlapped and regular sockets*

Technical Presentation

Requirements Summary

New Features by Category

- *Simultaneous access to multiple transport providers*
- *Latency and throughput enhancements*
- *Quality of Service*
- *Generic functionality extensions*

➢ *Open Issues*

Open Issues

➢ *Do we need scatter/gather?*

➢ *Should **WSASocket()** take a PROTOCOL_INFO struct as input?*

➢ *Do we need independent notification on shared sockets?*

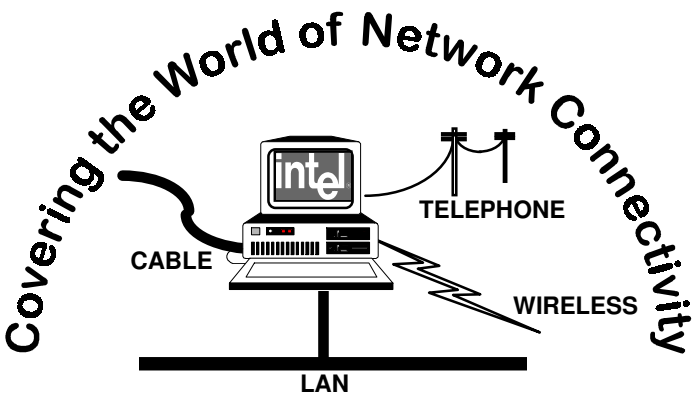
Open Issues (cont)

- Should our two groups be merged?

Next Steps

- Collect your comments and feedback
- Update the API and SPI docs
  - close as many “opens” as possible
- Publish docs, slides, meeting minutes
  - via anonymous ftp and the web
  - done by Christmas
- Prepare for presentation to WS2 Review Boards in January

Q&A / Feedback



Winsock 2