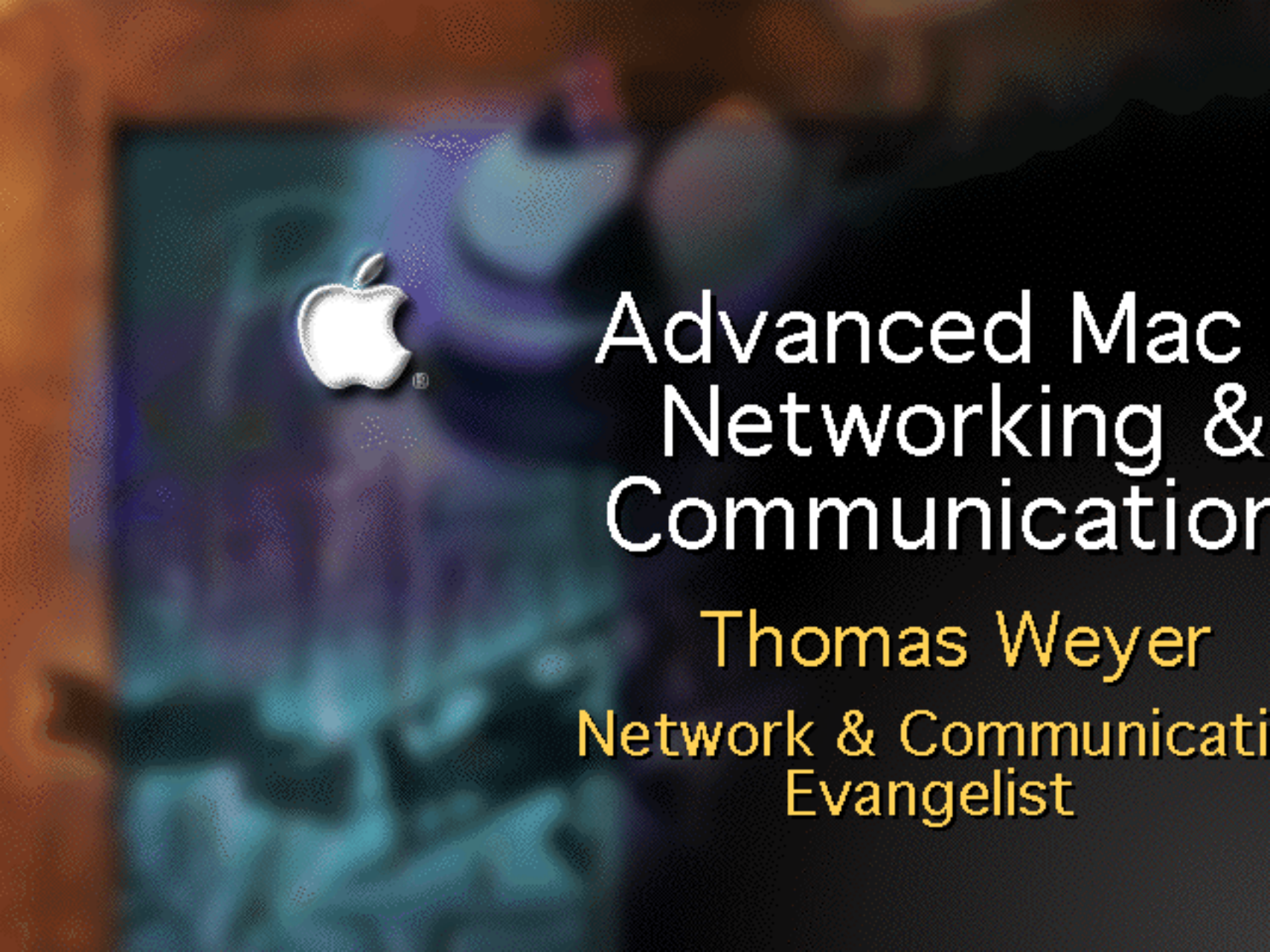

Worldwide
Developers
Conference


Advanced Mac
Networking &
Communicatio
Thomas Weyer
Network & Communicati
Evangelist

Advanced Mac OS Networki and Communications

Session 109, Thurs. 9:50-10:50 am, Ro

- Introduction and Logistics
- Session Overview
- Other Sessions of Interest
- Contact Information

Session Overview

- Introduction
  - Thomas Weyer (weyer@apple.com)
- Single Link multihoming
  - David Kim (davidkim@apple.com)
  - Chin Niazi (cniazi@apple.com)
- OT 1.1.x API Changes
  - John Comiskey (comisk.j@apple.com)
- Performance Optimization
  - Quinn (eskimo1@apple.com)
- Q&A
  - DTS & OT Engineering

Previous Sessions of Interes

Mac OS Networking & Communications

- Session 108, Wed. 5:50-6:50 pm, R
- Agenda:
  - Evangelism Introduction
    - Thomas Weyer (weyer@apple.com)
  - OT Roadmap and update
    - Richard Ford (rvf@apple.com)
  - ARA & OT PPP Roadmap and update
    - Richard Ford (rvf@apple.com)
  - Q & A
    - DTS & OT Engineering

Related Sessions

- Rhapsody Core OS Kernel and Runti
  - Session 201, Mon—4:50-5:50 pm, H
- Mac OS Support in Rhapsody's Blue
  - Session 204, Wed—1:30-2:50 pm, H
- Rhapsody Network Administration M
  - Session 221, Fri—4:30-5:30 pm, Roo
- Rhapsody Networking APIs & Servic
  - Session 220, Fri—5:50-6:50 pm, Roo

Related Sessions

- Feedback Forums
- Rhapsody Core OS Feedback Forum
  - Session 292, Thurs.—11:10-12:10 p
    Room J-4
- Mac OS Networking Feedback Forur
  - Session 194, Thurs.—1:50-2:50 pm,

Important Contact Informat

- ARA 3.0
  - Blanket NDA Seeding
  - CCL submission deadline of June 1st (msg@apple.com)
- Third-party OT mailing list
  - http://www.stairways.com/mailinglists/opentpt.html
- Apple Feedback addresses
  - opentpt@seeding.apple.com
  - rhapsody-dev-feedback@apple.com

Your Evangelist !!!

Thomas Weyer
N&C Evangelism
Apple Computer, Inc.
3 Infinite Loop, MS 303-2E
Cupertino, CA 95014
Phone: (408) 974-5017
Fax: (408) 974-1211
Email: weyer@apple.com
http://www.devworld.apple.
dev/opentransport/


Single Link
Multihoming
David Kim
Senior Software Engine
MacOS Networking

Introduction

- Ability to configure additional IP Secondary addresses with IP Primar address active
- Meet needs exhibited by ISPs prima for the purpose of Vanity IP addres multiple hosting by Internet servers
- Not for End User

Configuration

- IP Primary address is configured usi
  TCP/IP control panel
- IP Secondary addresses are configu
  using entries in a text file named "I
  `Secondary Addresses`", which
  resides in the System Preferences f
- Addresses are not validated, duplica
  addresses will be reported
- The number of addresses is only lim
  by available memory

Support

- New API to retrieve IP Secondary addresses
  - `pascal OSStatus OTInetGetSecondaryAddresses (InetHost* addr, UInt32* count, SInt32 val);`
- Primary address must be configure Manually

Support

- IP Secondary addresses:
  - Do not have to be configured in a contiguous block
  - Will use the same subnet as the Primary address

IP Secondary Addresses Format

- One IP address per line
- Example:

```
;list of IP Secondary addresses
15.0.0.2
15.1.1.5
15.3.4.5
;15.1.2.0
```

- Use of ";" on a line indicates the lin
  ignored, as in Hosts file

Developer Tips

- New `fIPSecondaryCount` field is adde to `InetInterfaceInfo` structure
- Call `OTInetGetInterfaceInfo` for the `fIPSecondaryCount`
- Call `OTInetGetSecondaryAddresses` for all the IP Secondary addresses
- Call `OTBind` with a specific IP addres listen on that address


Demo


OT API Change
John Comiskey
Senior Software Engine
MacOS Networking

Overview

- These APIs are new in OT 1.1.1.
- This is a subset, refer to the Develo
  Notes for more complete informatio

New DLPI Template

- `dlpiether.c` is a new DLPI driver template from Mentat that replaces previous template from Apple
- Supports version 2 of the DLPI spec
- Included in SDK along with detailed documentation

OTSetMemoryLimits and OTSetServerMode

- Force OT memory pool to grow immediately and never shrink
- Useful for servers that don't want t for the pool to grow, which can only happen at System Task time
- Prototypes for `OTSetMemoryLimits` a `OTSetServerMode` are not in any head files to prevent them from being ab

OTSetMemoryLimits

- `OSStatus OTSetMemoryLimits(size_t growSize, size_t maxSize);`
  - Pool will immediately grow by growS bytes and will never shrink
  - The maximum pool size is set to max
  - Default maxSize for OT is 10% of physical memory

OTSetServerMode

- `void OTSetServerMode();`
  - Pool will immediately grow by 2MB a
    will never shrink
  - Does not change the maximum pool

OTUseSyncIdleEvents

- Used to enable or disable `kOTSyncIdleEvent` for your endpoint
- When enabled your notifier will be called with a `kOTSyncIdleEvent` repeatedly while OT is waiting for a sync call to complete
- It is safe to call `YieldToAnyThread` a this point since it is only called at S Task time. This is the only notifier e that can safely be used to call `YieldToAnyThread`

OTUseSyncIdleEvents

- The notifier for the `ProviderRef` will receive `kOTSyncIdleEvent` while the `ProviderRef` is blocked by OT
- `OSStatus OTUseSyncIdleEvents(ProviderRef ref, Boolean useEvents);`
  - ref is the `ProviderRef` (i.e., endpoin that wants to receive `kOTSyncIdleEvent`
  - useEvents is true to enable `kOTSyncIdleEvent`, false to disable i

OTEnterNotifier and OTLeaveNotifier

- Used to prevent your notifier from being entered
- Useful for critical sections outside of the notifier

OTEnterNotifier and OTLeaveNotifier

- `Boolean OTEnterNotifier(ProviderRef ref);`
  - returns true if successful
  - returns false if the notifier is currently entered
- `void OTLeaveNotifier(ProviderRef ref);`

OTGate API

- Used to serialize the the processing data or events
- Common use is to serialize OT notif events across multiple endpoints to critical section issues
- Consists of `OTInitGate`, `OTEnterGate` and `OTLeaveGate`
- See “Open Tpt Module Dev. Note”

OTGate API

- Client example:
  - Serialize OT events with API calls tha your code handles from above
  - HTTP implemented as an OT client would need to serialize OT events wi calls to its API
- Module example:
  - When the sync queue level is less restrictive than `SQLVL_MODULE`


Open Transpor
Performance
Quinn “The Eskimo!”
Developer Technical Supp

Overview

- Why you *don't* need this talk!
- General rules for optimization
- Measuring your performance
- OT-specific optimization tips
- General Mac OS optimization tips

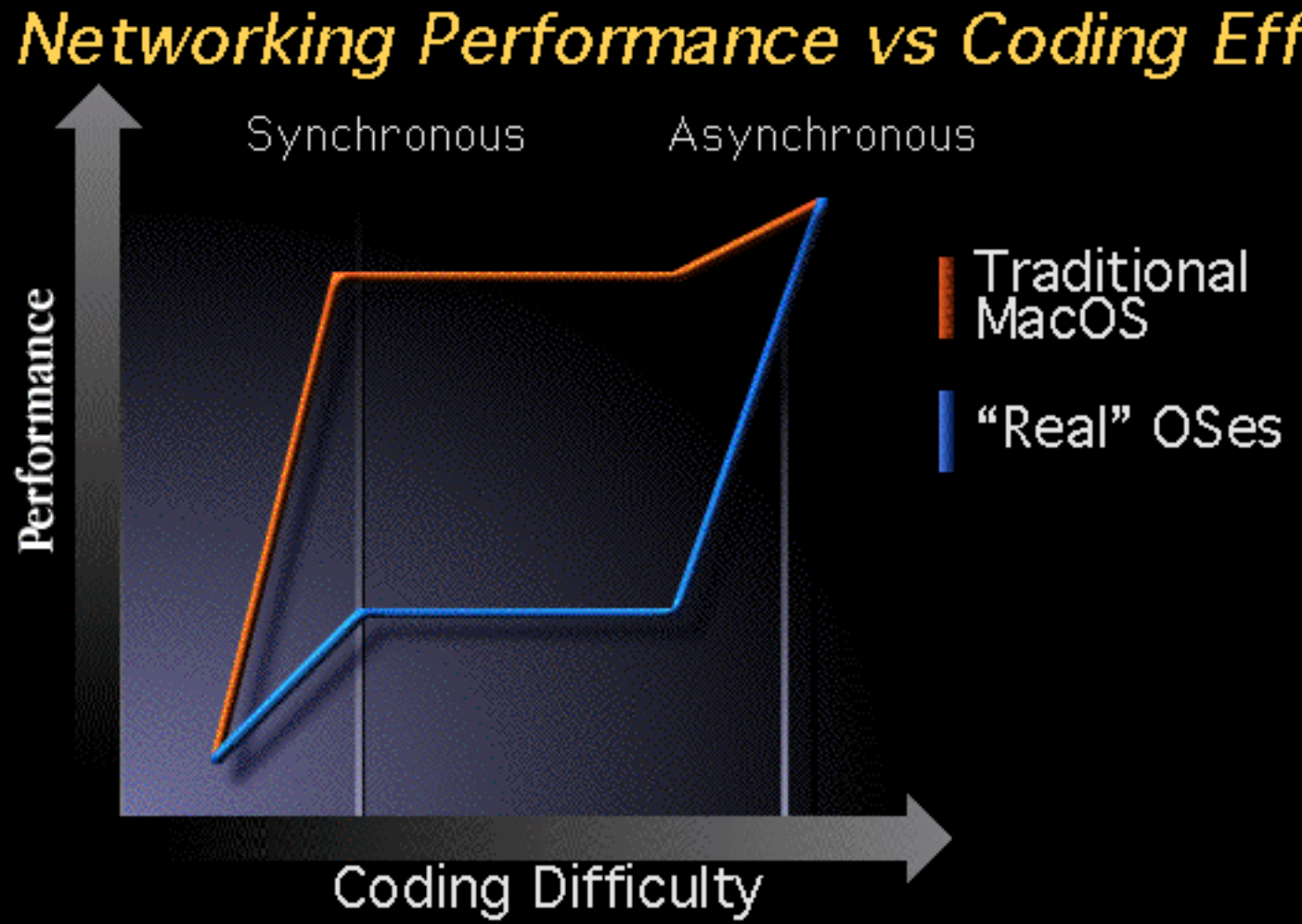
Why You Don't Need This T

Avoid Premature Optimization

- As a DTS engineer, I see this proble the time
- Bo3b's Rule: I'd rather be windsurfi
- It's hard to get 100% performance traditional Mac OS
  - So don't try unless you are 100% ce that your product needs it to succee
  - e.g., NewsWatcher, a very successfu Internet application, uses polled I/O

Quinn's Bogus Chart

Networking Performance vs Coding Effo



General Optimization Tenets

- Make it work, then make it go fast
  - Reliability is more important than spe
- Algorithms have the most effect
  - If you control both ends of the conn design your protocol for speed
  - Unfortunately, this is often not the c
- Decide on good performance metric
- Set a performance goal before you
- Stop when you hit it
- Profile before you change code

Algorithmic Examples

Change Your Approach, Go Much Faster

- AppleShare IP
  - You could tweak the small things unt end of time, but you would never ge speed benefits that switching to a fa underlying protocol yields
- FTP client
  - Using polled I/O for the command connection, notifiers for the data connections

Performance Metrics

- Highly application specific
- Maybe your program has an industry standard
  - e.g., WebStones?
- Otherwise, you have to develop your own metrics
- Don’t forget subjective metrics

Profile Before You Start

- Otherwise you risk wasting time optimizing the wrong thing
  - e.g., server might be file system bou network bound—if it's file system bo you're in the wrong talk!
- Instrumentation SDK
  - Allows you to add trace points to yo
  - Unlike normal profilers, these trace p aren't based on routine entry/exit
  - Works well for asynch I/O environme

Optimizing Your Network Co

- The next few slides contain specific suggestions
- Each suggestion has a corresponding caution
- You must decide which optimization are suitable for your application

OT Optimization #1

Program Structure

- Do everything in your notifier...Why
  - `WaitNextEvent` takes forever to ret
    - 10 ticks *is* forever on a 100Mbps netw
  - Notifiers offer important re-entrancy guarantees
  - OT offers interrupt-safe memory management
- However, there are alternative prog structures that may be fast enough

Caution #1

Program Structure

- The listen/accept hand-off sequenc for OT is particularly scary
  - See Technote 1059 for details
  - The `tilisten` module helps
- You must also take care to avoid `kOTStateChangedErr`
  - Use `OTEnterNotifier` to sync foreground and notifier threads
  - Wait for completion events before starting new operations

OT Optimization #2

Cache and Reuse Endpoints

- It takes longer to create/destroy en
  than to bind/unbind them
- Thus, if your program uses a lot of
  endpoints (e.g., a web server), you
  performance gains by keeping a cac
  unbound worker endpoints
- Don't bind worker endpoints, `OTAcc`
  will do that for you

Caution #2

Cache and Reuse Endpoints

- This technique has very few drawba
- You may find situations where the `OTUnbind` fails—in that case just clo and reopen the endpoint
- Also remember that each endpoint consumes memory that might be b used elsewhere

OT Optimization #3

Avoid Copying Data

- OT is very efficient about data copy
  - In the best case, OT can receive and transmit without any copying
  - But you must explicitly enable this a
- No-copy `OTRcv`
  - Returns pointer to OT buffers, you m return it with `OTReleaseBuffer`
- Ack Sends mode
  - You must keep data around until `T_MEMORYRELEASED`

Caution #3

Avoid Copying Data

- For no-copy receives, make sure yo return the receive buffer to OT as c as possible
  - In many cases, this buffer will be the driver's DMA buffer
- Ergo, don't modify the buffer!

Caution #3 (cont.)

Avoid Copying Data

- Ack Sends will complicate your cod
  because you must track memory
  allocation yourself
  - e.g., calling CloseOpenTransport with
    outstanding Ack Sends is fatal
- Under certain circumstances, OT m
  modify your data buffer

OT Optimization #4

Use OT Utilities Where You Can

- OT LIFO lists are better than the traditional OSUtils `Enqueue`/`Dequeue` routines
- OT's memory management is optimized for networking
- `OTTimeStamp` is the best timestamping on the traditional Ma
- OT's deferred tasks will adjust to system changes better than `DTInst`

Caution #4

Use OT Utilities Where You Can

- No caution needed. Just do it!

OT Optimization #5

Server Mode

- `OTSetMemoryLimits` allows you to increase the amount of memory OT use for buffers
- Useful for servers that are fielding l simultaneous connections
- If you just want to push the maximu number of bits through a *single pip* most probably won't help

Caution #5

Server Mode

- `SetMemoryLimits` is a case where a performance metric is very useful
  - Giving more memory to OT may improve your performance
  - ...or it may do nothing
  - You need to measure it for your particular application
- Definitely not recommended for client software

OT Optimization #6

Protocol Tweaks

- `XTI_RCVBUF`
  - Alters the endpoint's receive buffer size
  - Affects the TCP "window size"
- `XTI_SNDBUF`
  - Alters the endpoint's send buffer size
- TCP protocol has a lot of (currently private) knobs you can twiddle

Caution #6

Protocol Tweaks

- Like `SetMemoryLimits`, these techniques require intelligent use
- Changing these values can improve worsen, or have no effect
- The effect may even vary from site
- My recommendations:
  - Test in production environments to f sensible defaults
  - If you find great variance, put it unde user control

General Mac OS Optimizatio

- Try to avoid the file system
  - Cache everything you can
  - When you can't, call the file system asynchronously
  - And call it for big chunks of data
  - Double buffer file and network I/O
  - See DTS Technote FL 16 "File Manag Performance and Caching"
- Avoid taking MixedMode hits
  - Don't disable interrupts

Virtual Server Sample

- HTTP-like sample written by the OT TCP/IP lead
- Approximately 400 connections pe second on 200 MHz 604
- Very useful sample if you need spe
- Shows good techniques for most o optimizations in this talk
- Also shows how to workaround a nu of OT 1.1.x bugs

Wrap Up

- Think before you optimize
- Define your performance goals befo
  you begin
- OT can be very fast, if you're prepa
  do the work
- OT Virtual Server is the place to sta

Suggested Reading

- OT Web Page
  - http://devworld.apple.com/dev/opentransport/
- Inside Macintosh: Open Transport
- Technotes
  - Technote 1059 “On Improving Open Transport Network Server Performan
  - Technote FL 16 “File Manager Perfor and Caching”
- Instrumentation SDK—ETO 23


Q&A


Worldwide
Developers
Conference