Transport Protocols for Optical Burst Switched Networks

Moving Beyond Lightpaths

Arnold Bragg
Advanced Networking Research Division
MCNC Research and Development Institute
Research Triangle Park, NC 27709 USA
abragg@anr.mcnc.org

Acknowledgments

- Co-authors and contributors:

Ilia Baldine	Joel Hernandez	Dan Stevenson
Arnold Bragg	Bonnie Hurst	Steve Thorpe
Stephanie Bryant	Gigi Karmous-Edwards	Raghu Uppali
Mark Cassada	Mike Pratt	Xiaoyong Wu

- Sponsors:
  - NASA under award NAG2-1467
  - The US Intelligence Community under ARDA contracts MDA 904-00-C-2133 and MDA 904-02-C-0482

- Disclaimer:
  - "Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsor(s)."

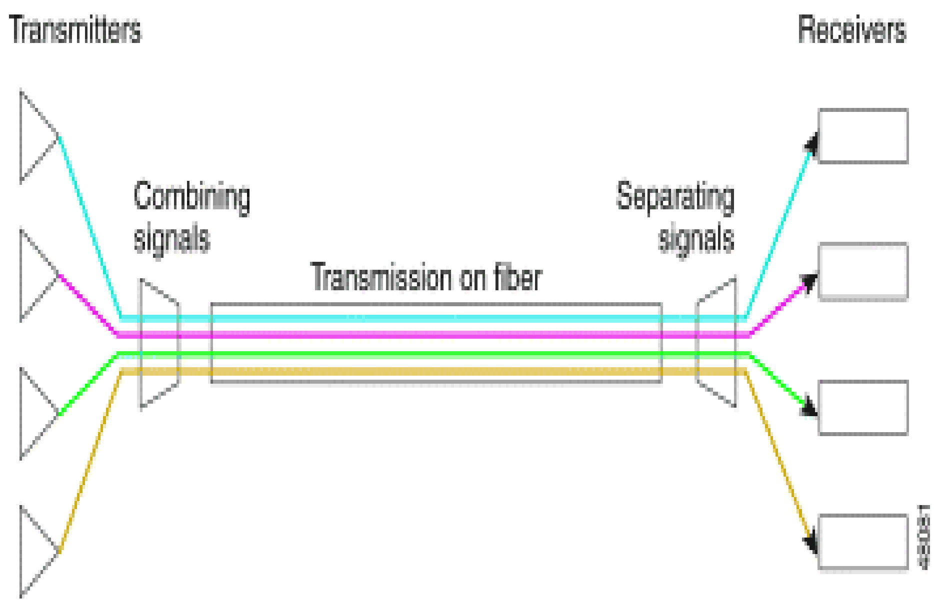
Topics

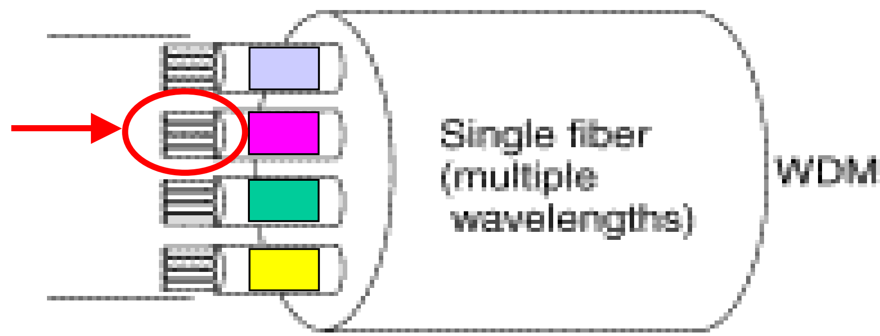
- What is optical burst switching?
- What does OBS have to do with fast, long distance networks?
- What's been done?
- Main points

What is optical burst switching?

What is optical burst switching?

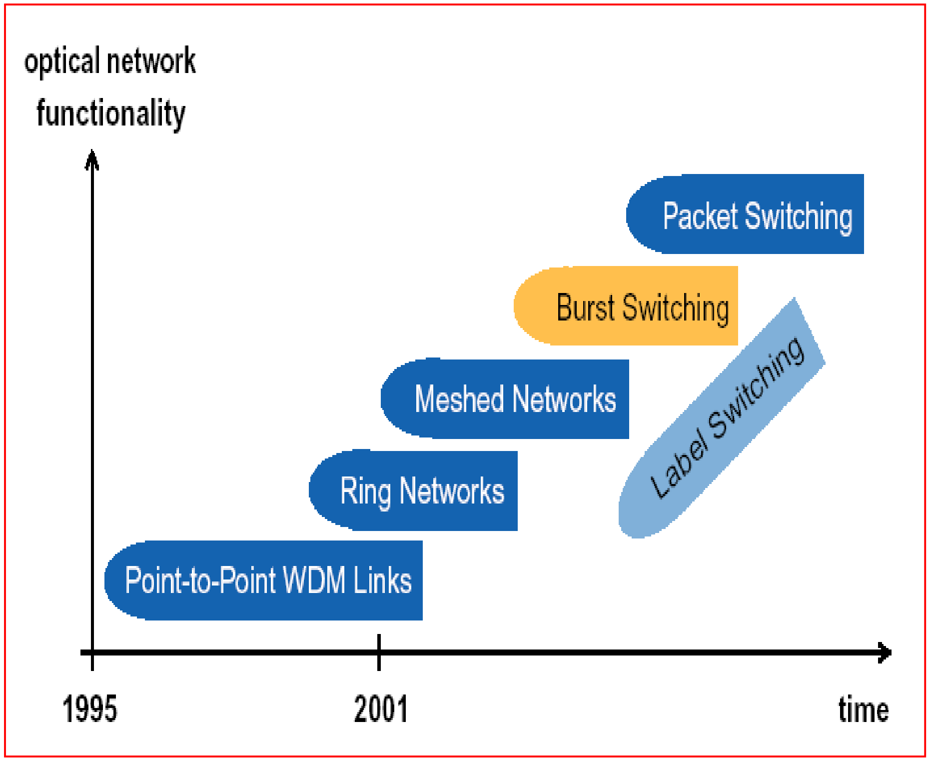
- WDM puts 10s to 100s of wavelength channels (λs) on a single optical fiber
  - A way to share links
  - Typically provision a λ for a source/destination pair
  - Can perhaps switch λs
- Add an OBS overlay
  - Provision λs for any duration
  - Switch and manage λs
  - Use features to greatly reduce contention & blocking
  - Share λs in time via fast provisioning & switching





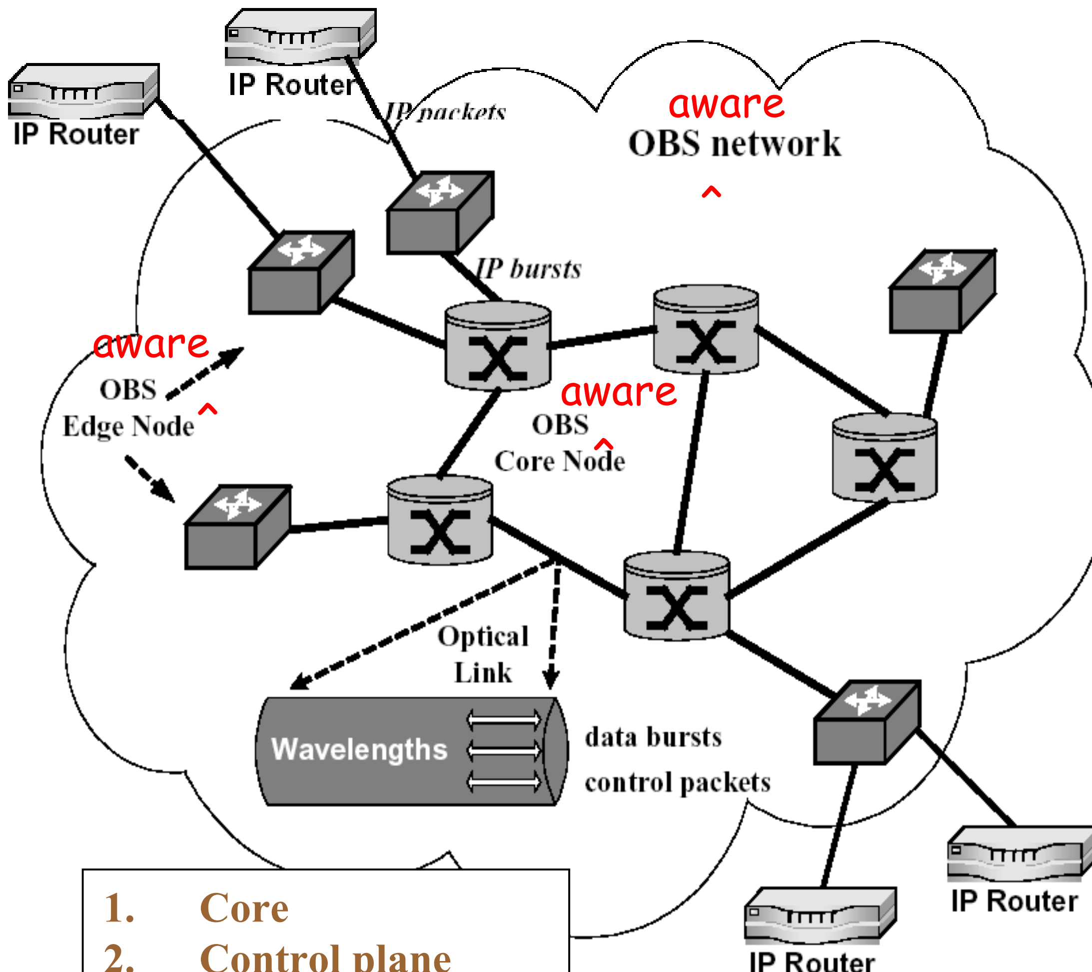
Provisioning and sharing λs

- **Provisioning** and **sharing** are important concepts
- Optical **circuit** switching
  - λ provisioning in minutes to months; long holding times
  - Unshared λ per s/d pair, or sharing via grooming or muxing
- Optical **packet** switching
  - λ provisioning in ns
  - Goal is all-optical; this requires optical buffers and optical header parsing
  - May be 10+ years out
- Optical **burst** switching
  - λ provisioning in ns, µs, ms
  - Short(er) holding times
  - No buffering in core
  - Header info out of band



Just a thin overlay …

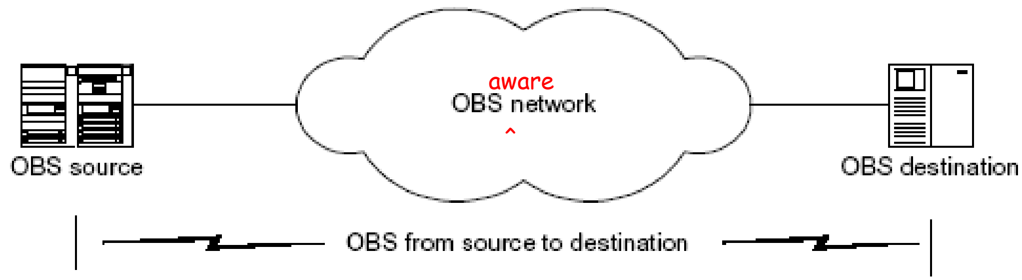
- OBS-aware networks
  - Data & control planes
  - Protocol agnostic
  - Subtending realms can be anything, including analog
  - Some architectures use COTS optical switching gear (OEO, OOO)
  - Control plane is a thin signaling overlay
  - Control plane can be implemented in h/w or in s/w
  - No global synchronization



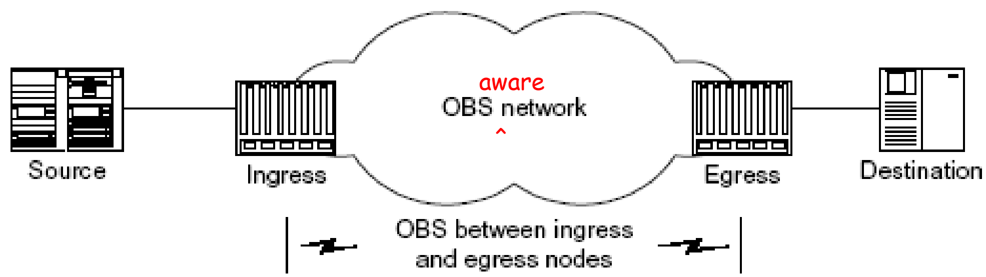
1. Core
2. Control plane
3. Management plane
4. Edge

Several scenarios

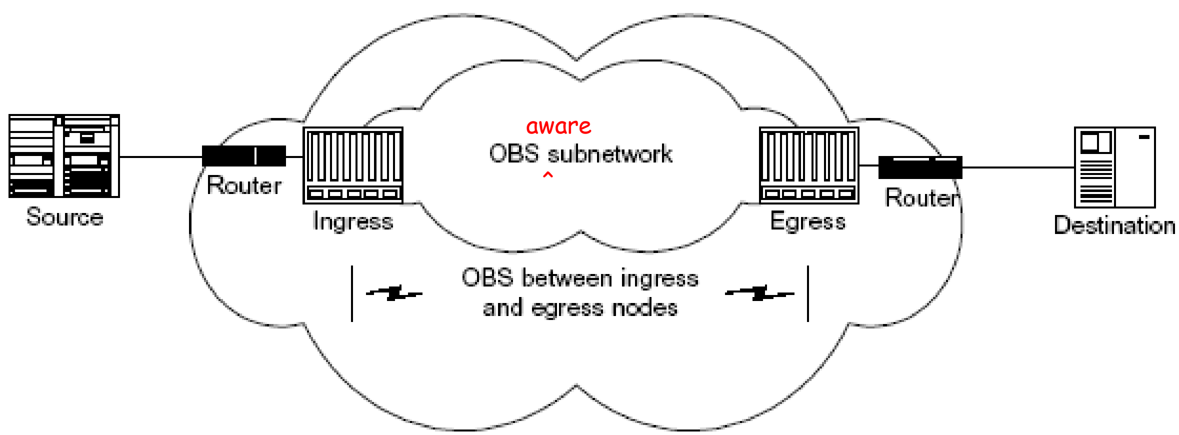
Scenario 1



Scenario 2

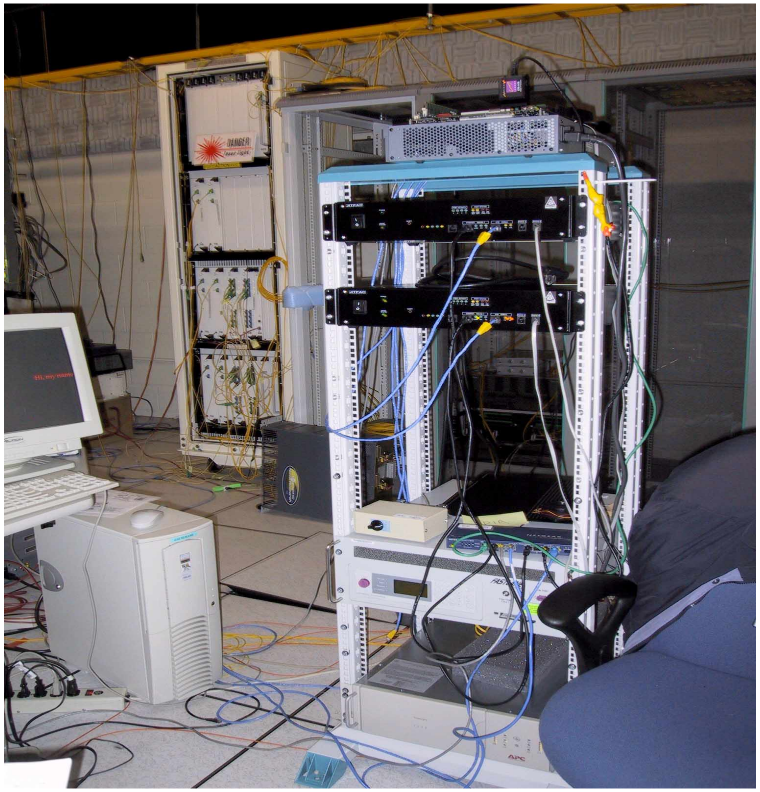


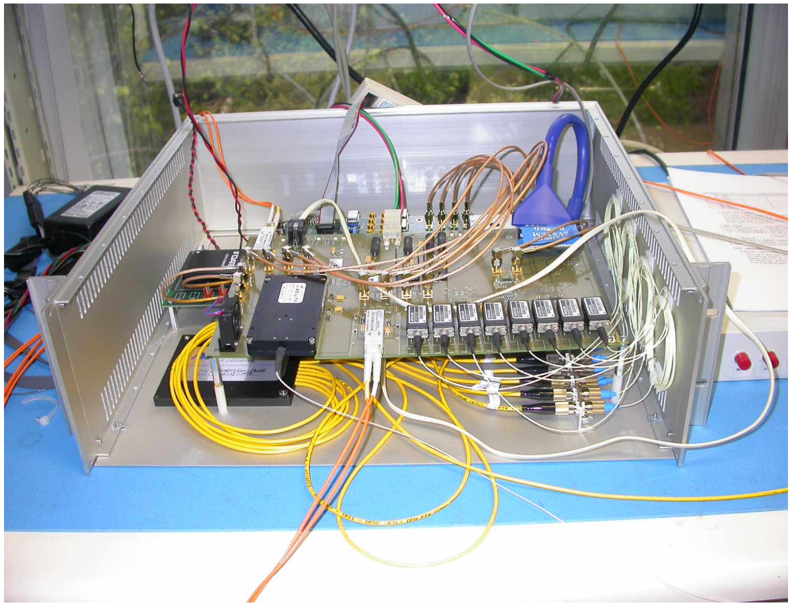
Scenario 3



Real or hype?

- Controller
  - Test bed deployment; all-optical COTS core switch in background;
  - OBS hardware controllers (v1) on rack in foreground
  - Software controllers have also been developed





- OBS-aware edge device
  - NIC, or aggregator, or OBS-aware host, or ...

What does OBS have to do with fast long distance networks?

What does OBS have to do with FLDNs?

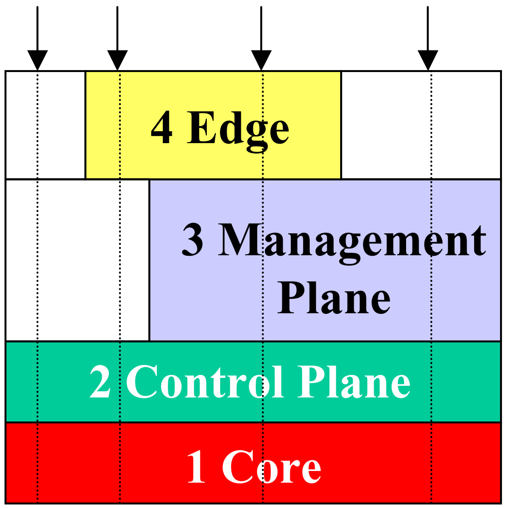
- 'Bored chameleon' nature of OBS is useful for many FLDN applications
  - Many shades of λ provisioning
    - Circuit -- rapidly provisioned, any duration
    - Tunnel -- intermittent traffic transiting a 'pinned route'
    - Packet(s) -- on-the-fly, per-burst routed; or flow routed
    - Anycast -- unicast, multicast, broadcast
  - Network core's data plane is unconcerned with payload, protocol, rate, format, encoding, modulation scheme, …
- Transport layer (i.e., L4) can take advantage of this
  - Some transport layer services are superfluous; e.g.,
    - OBS pinned routes guarantee sequenced delivery
    - OBS persistent paths guarantee zero (added) jitter
  - OBS-specific modifications can streamline transport protocols and protocol stacks

What does OBS have to do … ? (cont.)

- Performance
  - Today's dedicated circuits ('lightpaths') have some limitations
    - Scalability
      - A few tens of λ available even in DWDM networks
      - Lightpaths are usually not (rapidly) switched
    - Efficiency
      - Lightpaths hold, but rarely use, all of the bandwidth reserved for them
      - Most don't share bandwidth
  - Ultra fast provisioning/release of resources → more efficient sharing of bandwidth → multiplexing gain
  - With an OBS-aware edge device, you can shape traffic
    - To control or contain aggressive protocols
    - To provide fine-grained rate controls, pacing, …
    - Etc.

Keep it simple

- You don't need all four pieces
  - (4) Edge devices are optional
  - (3) Management plane is optional
    - Stateful overlay
    - Robust QoS-aware forwarding/routing
    - OAM, network management, etc.
  - (2) Control plane is required
    - Ultra fast provisioning
    - Fine-grained multiplexing via ultra short-lived lightpaths
    - Several flavors (bronze [s/w], silver/gold/platinum [h/w])
    - Inexpensive and unobtrusive overlay
  - (1) 'Core' is unmodified COTS gear
    - No forklift upgrades in the core; simple configuration



- (1) + (2) above are sufficient to provide ultra fast provisioning and fine-grained multiplexing

Performance

- So why not use GMPLS with RSVP or CR-LDP?
  - Slower provisioning, reliable signaling, longer holding times
- For ultra fast provisioning, short(er) holding times:
  - Simplex ("tell and go") OOB signaling; no multi-way handshake
  - Holding times on the order of milliseconds to hours
  - No "lambda tax"
  - Add an OBS edge if you need it (NIC, aggregator, aware host, …)
- Signaling performance*

Hardware v1	FPGA (Altera EP20K400)	~ 12.5 µs (80K setups/s)
Hardware v2	FPGA	~ 10x improvement
Hardware v2	ASIC	~ 100x improvement
Software	Commodity GHz PC	~ 3-10x slower

* Does not include switch configuration, transmission, propagation delays

What's been done?

Usual approach

- I work with a ___ network architecture
- In this architecture, my application is most efficient when the transport protocol is ___, so I'll use that

  OR

- I have a set of transport protocols, and I'll choose what's best in this architecture for my application
- One degree of freedom

  Application × network configuration × {transport protocols}

Core + control plane (1+2) approach

- I work with a quasi-configurable network, capable of fast provisioning/release, route pinning, fine-grained multiplexing, …
- My application is most efficient when the network is configured to appear (to my application) as a
    - Circuit - rapidly provisioned, any duration; or
    - Tunnel - pinned route pipe; or
    - Flow - with on-the-fly, per-burst forwarding; or …
- I'll choose a network configuration option
- I'll choose an efficient transport protocol for that configuration
- Two degrees of freedom

  Application × {network configurations} × {transport protocols}

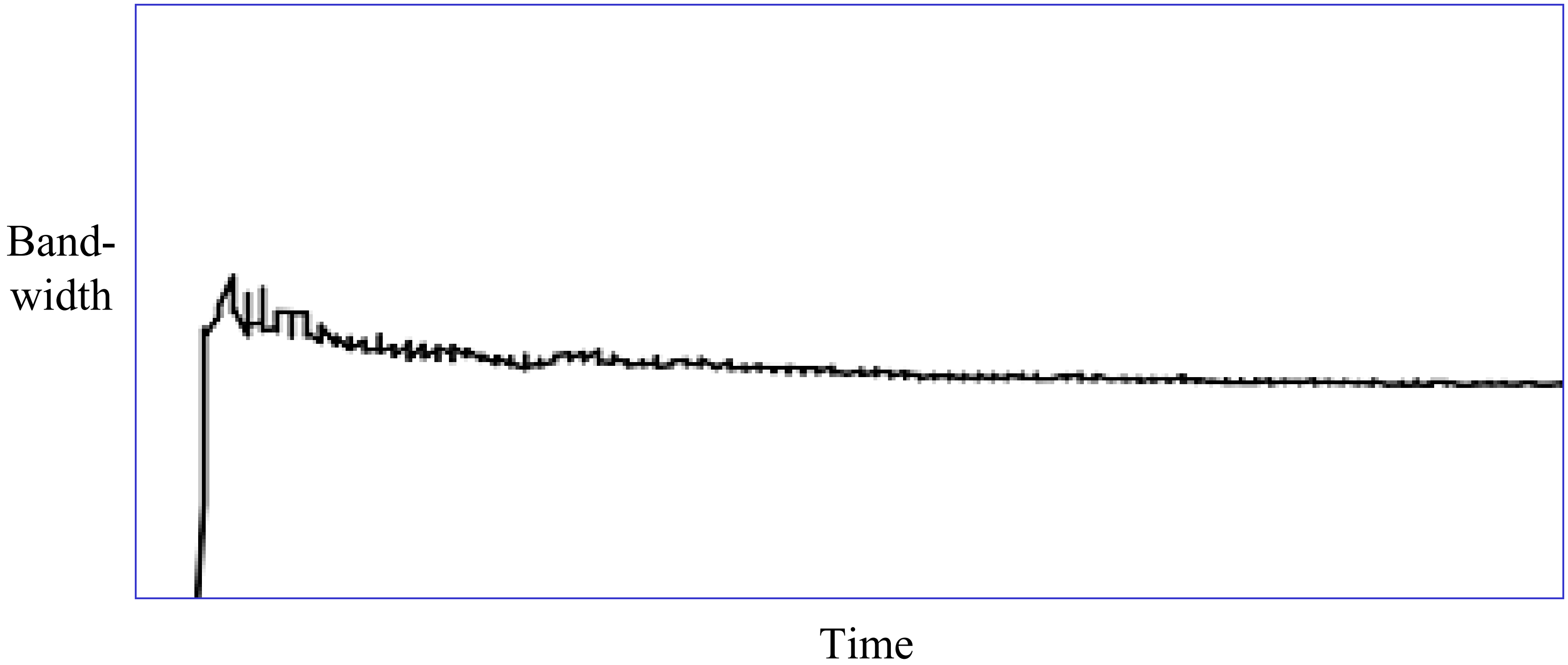
However …

- Some transport layer services are superfluous in configurable networks
- Why?
  - Performance dimension; e.g.,
    - No buffering in the core -- no queue delay/jitter/loss issues
    - Quick (μs) blocking indication -- no lengthy timeout intervals
    - Signaling is simplex -- no round-trip setup delay
    - Data follows signaling after a short head start, so there isn't even a one-way setup delay
  - Transport layer services dimension; e.g.,
    - Sequenced delivery service provided via route pinning
    - No jitter added in transit
    - Flows can be prioritized and preempted in the core
    - Can use the signaling channel for L4 ACKs, SACK, rates, etc.
    - Etc.

New approach

- My application is ___ and has these characteristics
  - I can provide them, or they can be inferred
- My application needs a network configuration of some type
  - I can choose, or my application can choose
- My application needs a transport protocol
  - I can choose, or my application can choose
  - I may not need all the services that the TP offers because the network configuration provides these services
- My application can choose from a reduced set of:
  - Feasible configuration and transport protocol combinations
  - Transport layer services
- So what?
  - (1) A-/u- initiated configuration; (2) 'tuned' TPs; (3) new TPs

Application-initiated network configuration

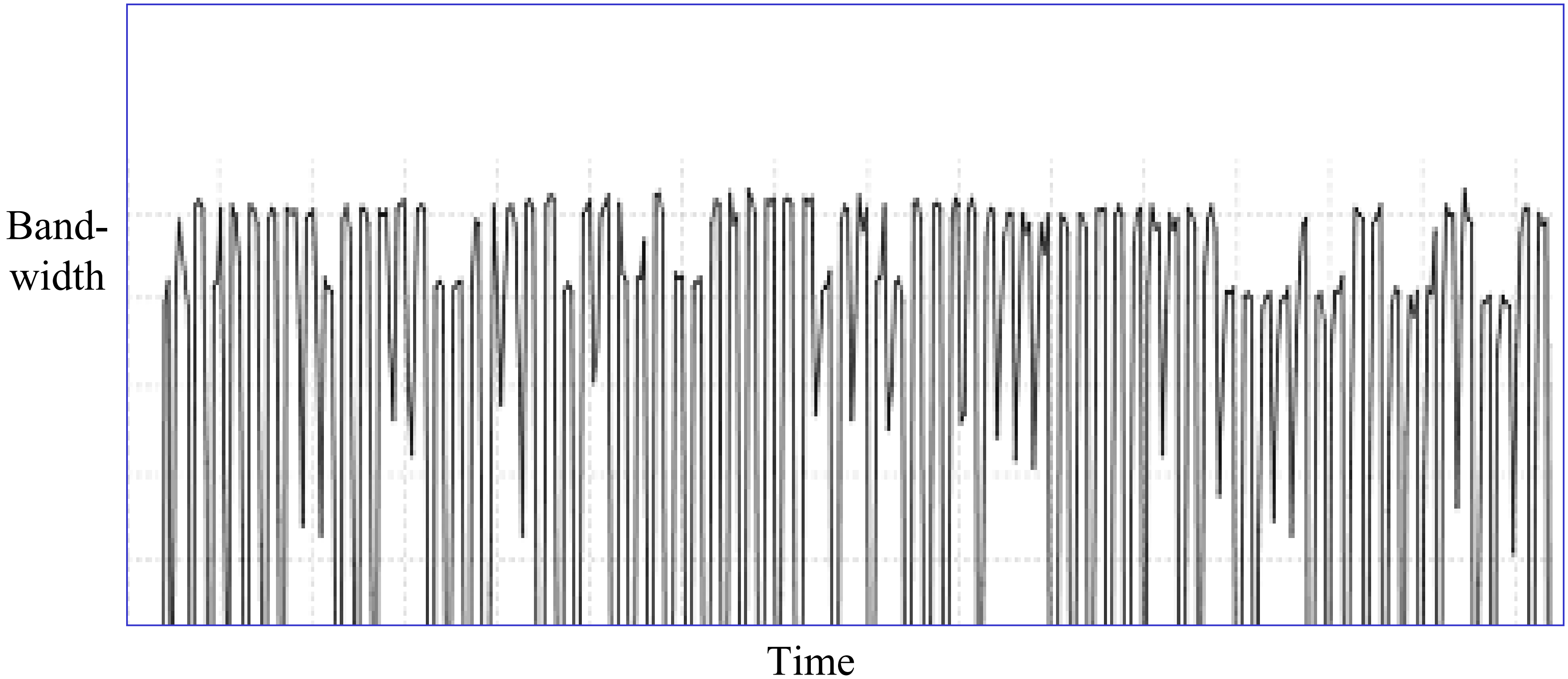


- Given the performance above, I'd like to:
  - Shorten provisioning time (at left)
  - Shorten release time (off scale at right)
  - Reallocate the unused bandwidth (at top)

Application-initiated network … (cont.)

- Team has developed an API for grid service clients
  - For user-/application-initiated provisioning
  - To provide improved performance -- scalability and efficiency
  - Supports application-initiated, GSI-authenticated, network connections via an OGSA interface
  - Client application is responsible for sending/receiving data once a connection has been provisioned
  - Requires relatively little information for provisioning
    - Addresses, paths
    - Timer intervals
    - Setup ACKs (y/n)
    - Explicit release of lightpath (y/n), etc.
  - Proof-of-concept stage
  - Deployment by 3Q 2004

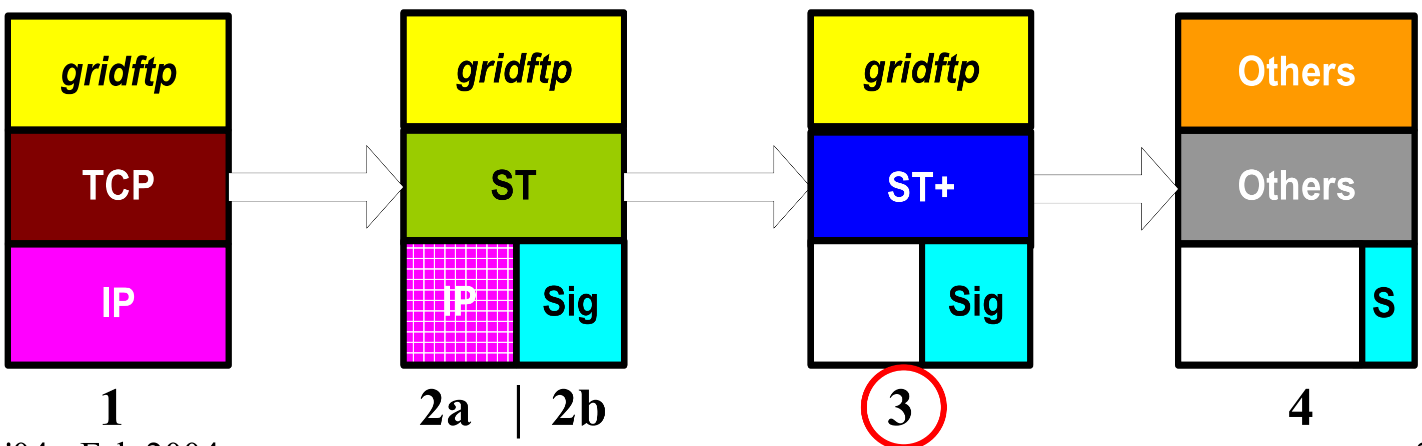
'Tuned' transport protocols



- Given the performance above, I'd like to:
  - Shorten provisioning and release times (left, right)
  - Reallocate the unused bandwidth (top)
  - Reallocate during dropouts (at bottom)

'Tuned' transport protocols (cont.)

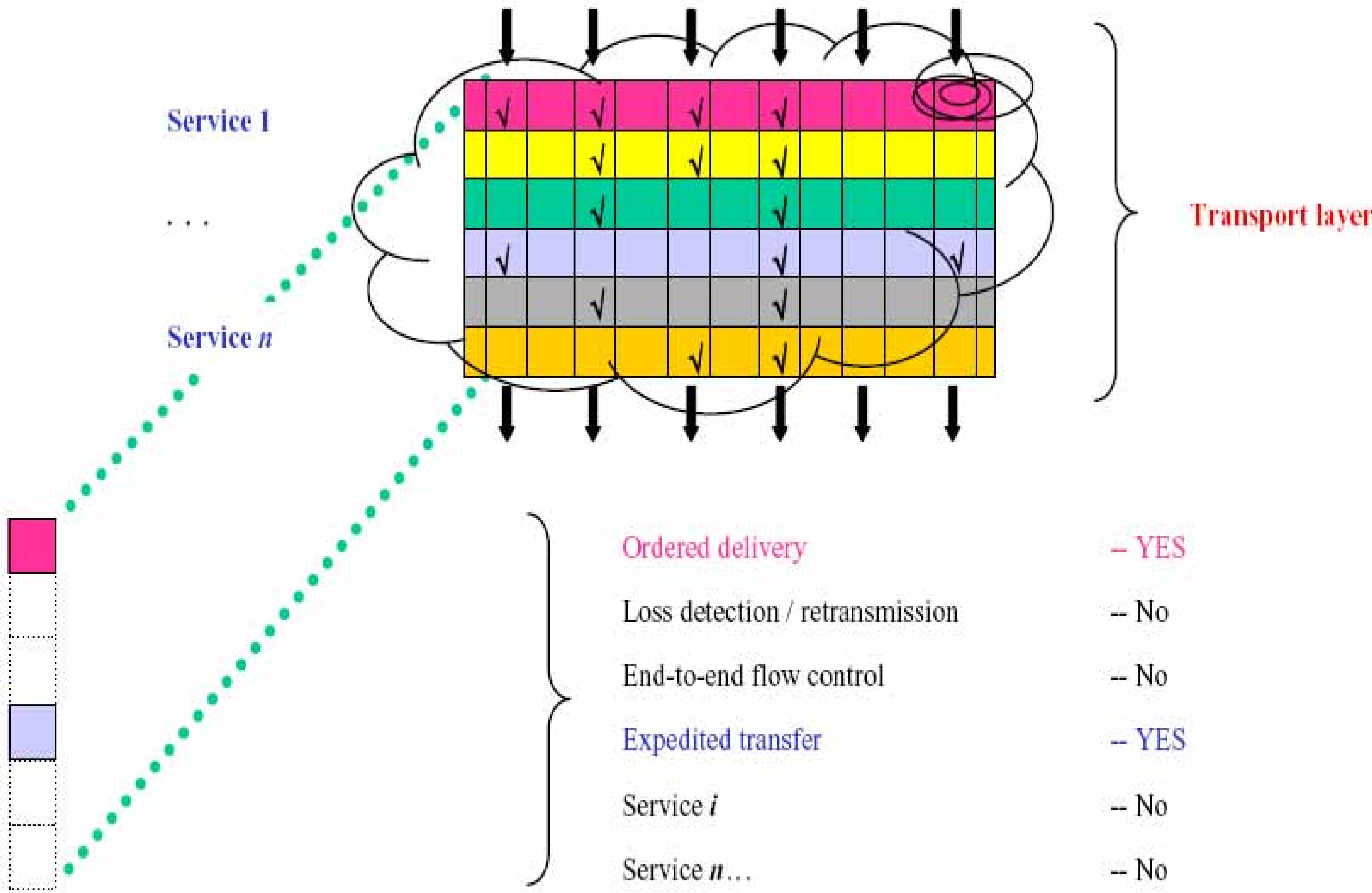
- Team has modified the Scheduled Transfer (ST) transport protocol (ANSI INCITS 337-2000)
  - To support application-initiated network connections via the TP
  - The modified TP (ST+) initiates signaling for data transfers as required
  - Implemented on SGI hosts with IRIX and Linux kernel mods
  - Testbed deployment with SGI hosts on paths transiting multiple OXCs over a 100 Km diameter network
  - Significantly leaner protocol stack



New protocols

- We said …
  - Some transport layer services are superfluous in easily configured networks
    - Performance dimension
    - Transport services dimension
  - My application … has these characteristics …
  - My application needs a network configuration of … type
  - My application needs a transport protocol …
    - I may not need all the services that the TP offers
  - My application can choose from a reduced set of:
    - Feasible configuration and transport protocol combinations
    - → Transport layer services
    - Implement some services in hardware
- Ideas about the design of new transport protocols operating in OBS-aware networks, or (1)+(2) networks

New protocols (cont.)



New protocols (cont.)

- Build on architectural features of quasi-configurable networks
- Adaptively monitor, so transport services vary so as to maintain QoS objectives in response to changing conditions
  - Mitigate retransmissions
  - Assert preemption and prioritization
- Burst assembler/scheduler and transport layer can work in concert to provide a number of useful services
- Use an OBS-aware edge device
  - Hardware-assisted rate and flow control, shaping
  - Varying degrees of determinism; e.g., burst by size (probabilistic delay bounds) or by time (deterministic)
  - Control or containment of aggressive protocols

Main points

Main points

1. Simple overlay for big science and other high performance networks
   - Works with unmodified COTS gear; simple to configure; agnostic
   - (1) + (2) sufficient for ultra fast provisioning and some muxing
   - Add (4), or (3) + (4) for ultra fine-grained multiplexing
   - Significant performance advantages
2. Application x {TPs} --> application x {TPs} x {netConfigs}
   - {TPs} x {netConfigs} has some important implications
   - Potential overlap between TP services and what network provides
   - TPs tuned to use net-provided services
   - New or modified TPs in which some services are handed off to the configurable network (edge and/or core)
3. Good fit for applications in FLDnets and grids
   - Performance advantages; chameleon nature
   - User- and application-initiated provisioning

Arnold Bragg

Advanced Networking Research Division

MCNC Research and Development Institute

Research Triangle Park, NC 27709 USA

abragg@anr.mcnc.org