The mash-up of virtual computation and virtual networking enables new paradigms in adaptive and disruption-tolerant IT

Franco Travostino
travos@nortel.com
Director, Advanced Technology and Research, Nortel
Area Director for Infrastructure, GGF

Virtualized Computation

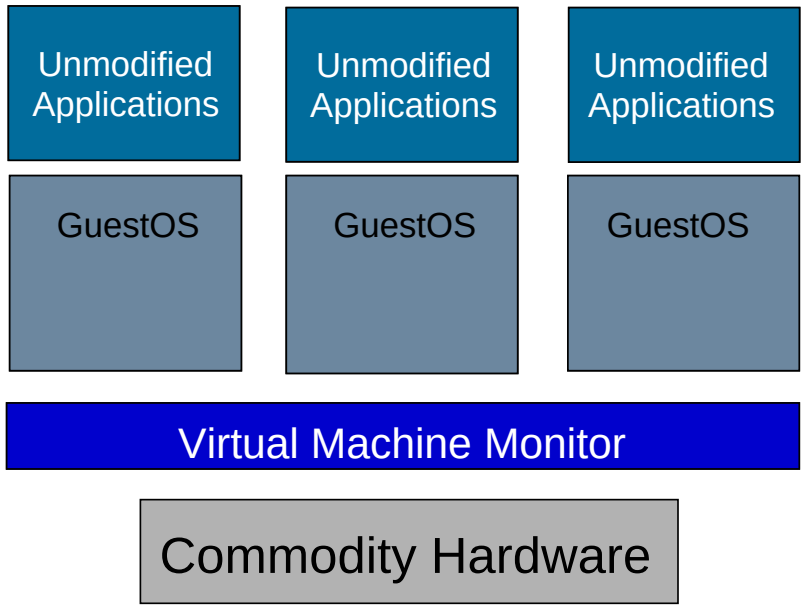
Virtualized Networking

Virtual Machines

the “new” workhorse of computation

- With VMs, multiple guest Operating Systems coexist on the same physical hardware, unbeknownst to one another

- A vision that came a long way
  - 1967: cp67 system for IBM 360
  - ’80s: Microkernels (Minix, Mach)
  - ’90s: the Java Virtual Machine
  - 1998: VMware
  - 2003: Xen’s “paravirtualization”
  - 2006: Intel’s VT-x, AMD Pacifica



Without HW cooperation, virtualization still is imperfect science

> Faulting and non-faulting to privileged state

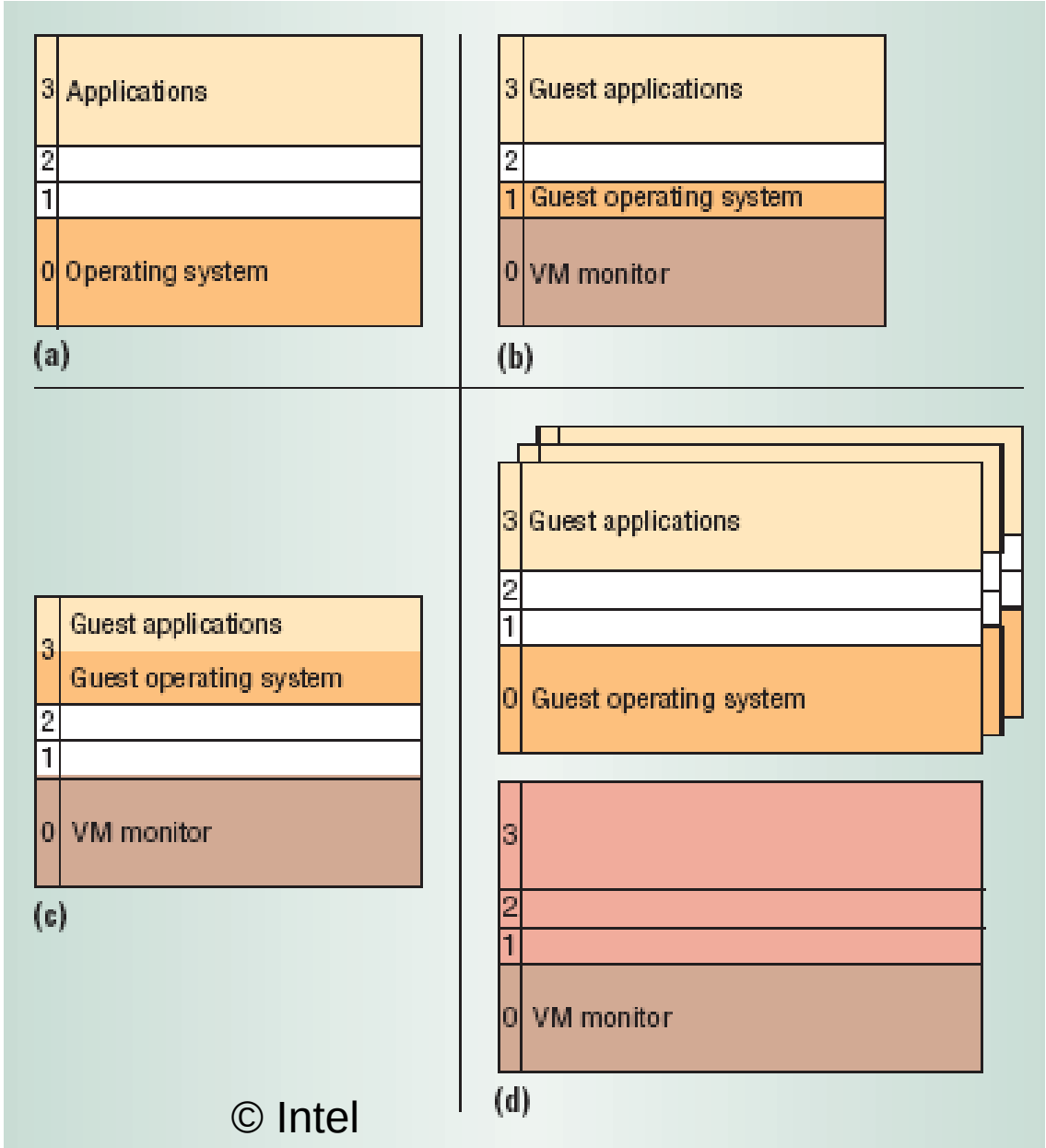
> Efficient (un)masking of interrupts

> Contiguous memory

> Compartmentalization of faults, attacks, etc.

> By popular demand, chip makers answered the call

> Intel's VT-x, AMD's Pacifica



A mainstream use of VMs

Isolation, consolidation for optimal sharing of infrastructure

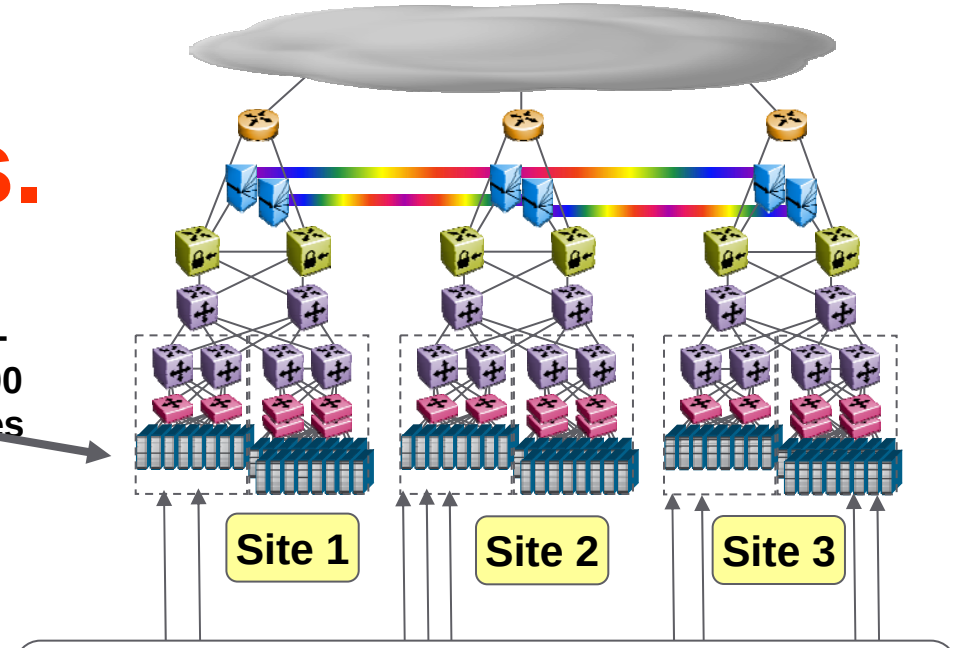
My Department thinks that the "Computons" spring out of here

w/ right-sized bandwidth, 24x7, at 100% disaster-free Zip code

VS.

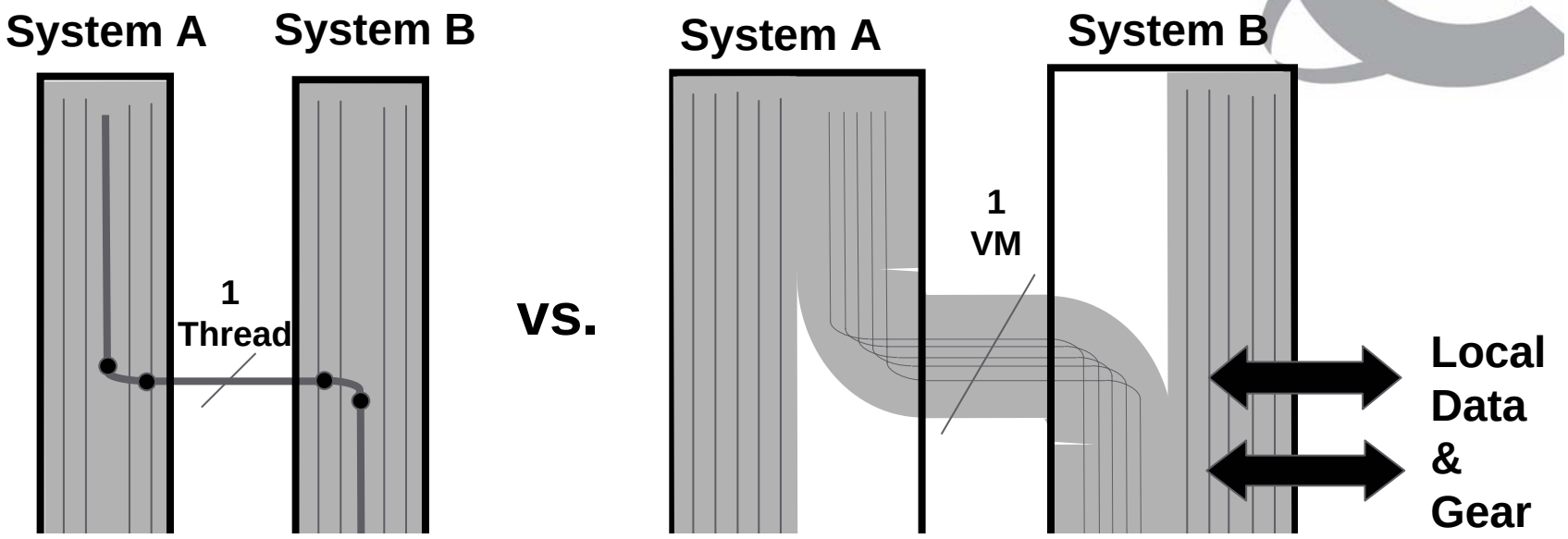
The way an IT organization operates for in-house or external accounts over a geographical footprint

100 - 10,000 blades



Franco's Virtual Machines run here w/ own Apps, Licenses, Lifecycle

VMs can Migrate



- Birrell and Nelson, RPC, 1984
- Clark et al., The Alpha Kernel, 1990
- Pratt et al., VM migration over LAN, ’04
- Travostino et. al., VM migration over MAN and WAN, ‘05

The case for migrating VMs outside a Data Center

Why would I take computation out of a Data Center:

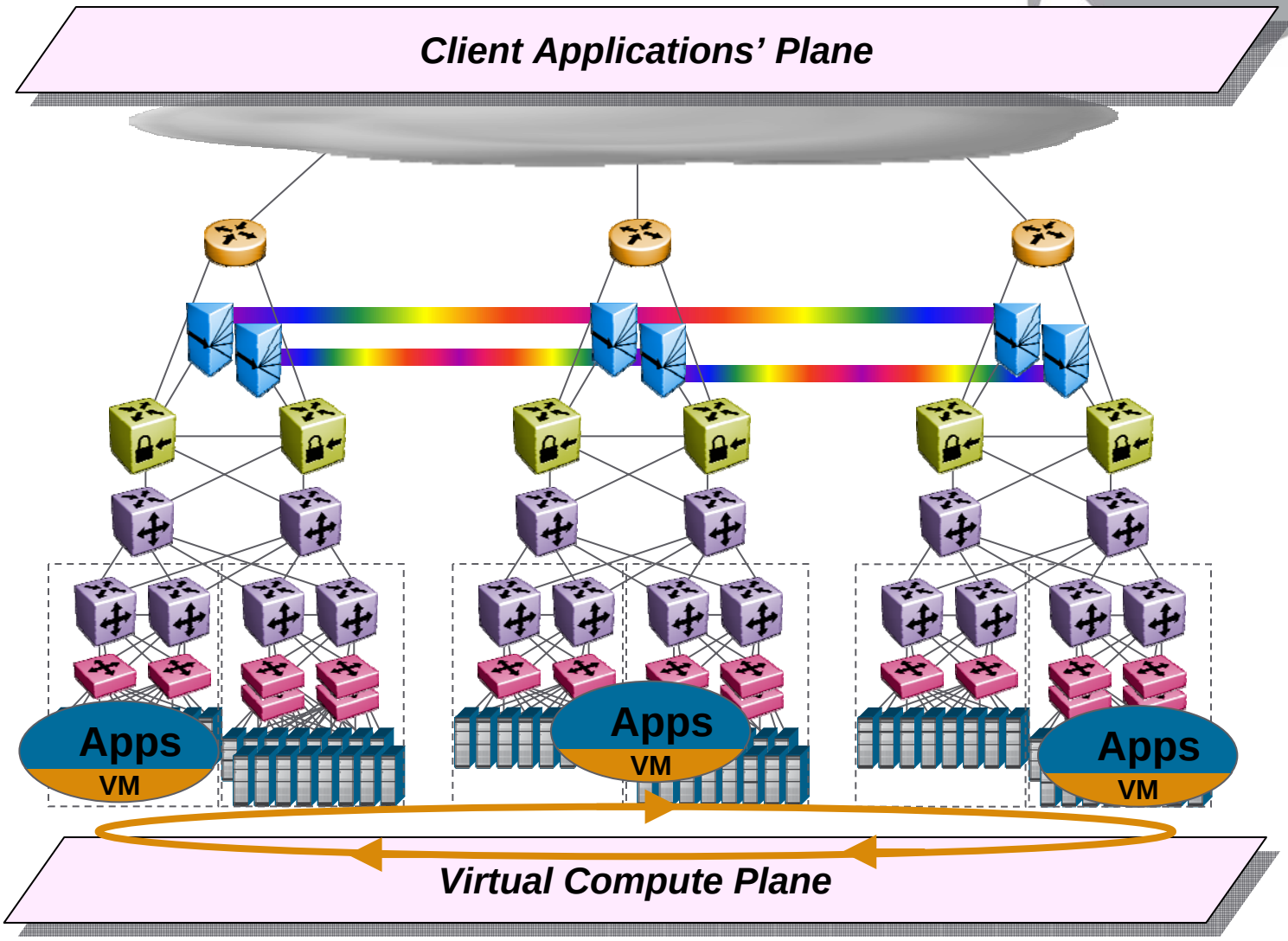
- > Cannot bring data (or devices) close to computation
  - Policy, size, workflow make it impractical
- > Want to spill over onto neighboring Data Centers
  - Pursuing OpEx, power savings, and “follow the moon” efficiencies
- > Business continuance, disaster recovery
  - e.g., to escape from failed, compromised, DoS-ed environments

Why would I use VMs for this:

- > Don’t trust remote execution environments
- > Curtail interoperability woes
- > Cannot alter/sever inter-application synapses, legacy applications

Virtualized Data Centers

the computation angle



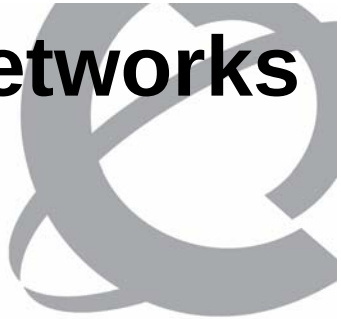
VM as a Service: Programming Model

- Instantiate a new VM
- P2V
- V2P
- Live migrate a VM to an endpoint
- Hibernate a VM
- Clone a VM
- Tear down a VM

Virtualized Computation

Virtualized Networking

Problems w/ unknowable hardwired networks



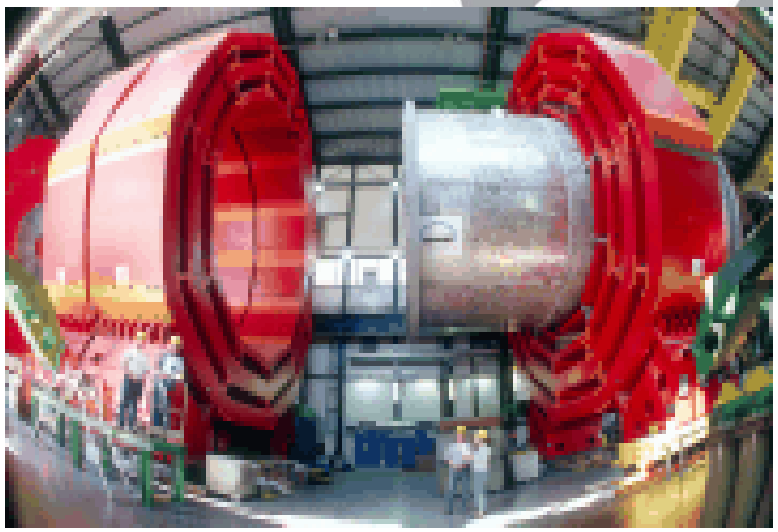
> Over-provisioning
  - “The only QoS that you can really depend on”
  - Especially across multiple domains

> With many unsatisfied users still
  - “Thou shall adapt the application to the network”
  - Only mainstream SLAs admitted
  - Lag before the supply side reflects new requirements in a SLA

> New apps urged to go global, yet without any perf compromise

> Network as a mirror of society: How unlikely that *everyone* plays by the rules
  - From TCP Daytona to (D)DoS

Contrasts



> Soon, Cern's LHC will post 10+ PB new data per year to researchers worldwide

vs.

> One size fits all networks

" a standard TCP connection with 1500-byte packets and a 100 ms round-trip time, achieving a steady-state throughput of 10 Gbps would require an average congestion window of 83,333 segments and a packet drop rate of, at most, one congestion event every 5,000,000,000 packets" (Floyd, '03)

Fat and fatter pipes ain't the (sole) answer

[...]

3.1.2 Abstraction/Virtualization

3.1.3 Site Autonomy

3.1.4 Flexibility/Programmability

3.1.5 Determinism

3.1.6 Decentralized Control

3.1.7 Dynamic Integration

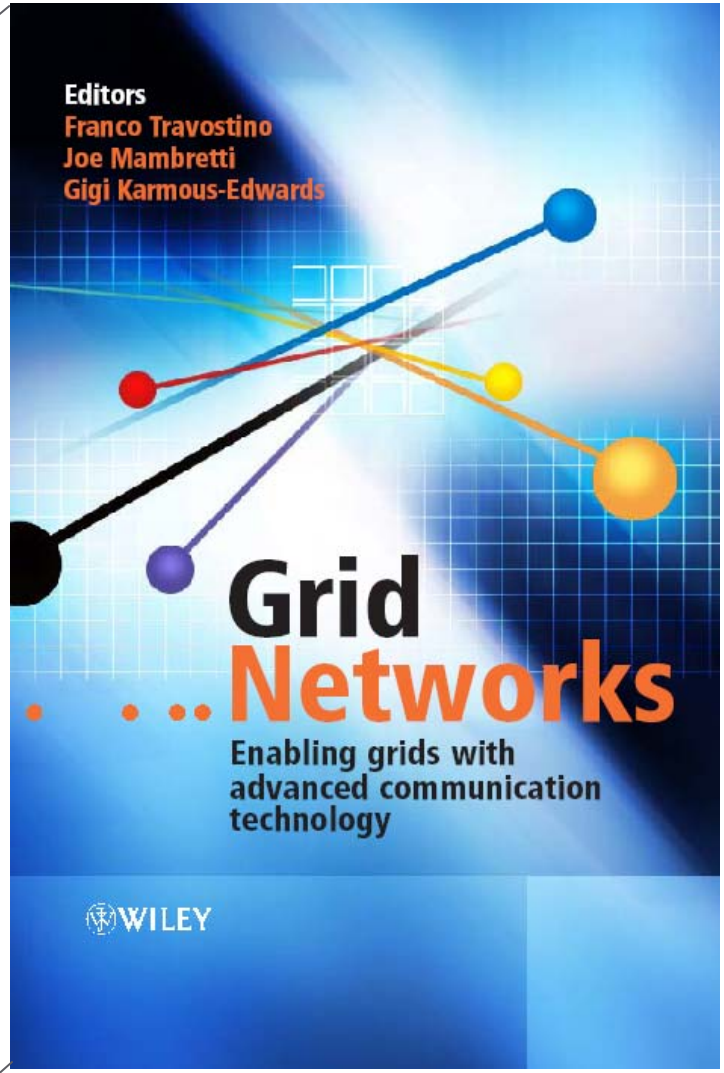
3.1.8 Resource Sharing

3.1.9 Scalability

3.1.10 High Performance

3.1.11 Isolation and Security

[...]



Available Summer 2006

Vision

> An application drives shares of network resources
  - Resources = {bandwidth, security, acceleration, policies, …}
  - Within policy-allowed envelopes, end-to-end
  - No more point-and-click interfaces, no operators' involved, etc.

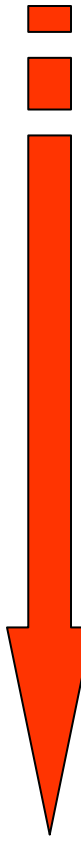
> Service-enable the network for greater control of such resources
  - Ex: JIT, TOD-schedulable control of bandwidth
  - Create alternatives to peak-provisioning across LAN/MAN/WAN
  - With a continuum of satisfaction vs. utilization fruition points

> Tame and exploit network diversity
  - Heterogeneous and independently managed network clouds, e2e
  - Ex: Integrated Packet-Optical to best match known traffic patterns

> Network as a 1st class resource in Grid-like constructs
  - Joins CPU, DATA resources

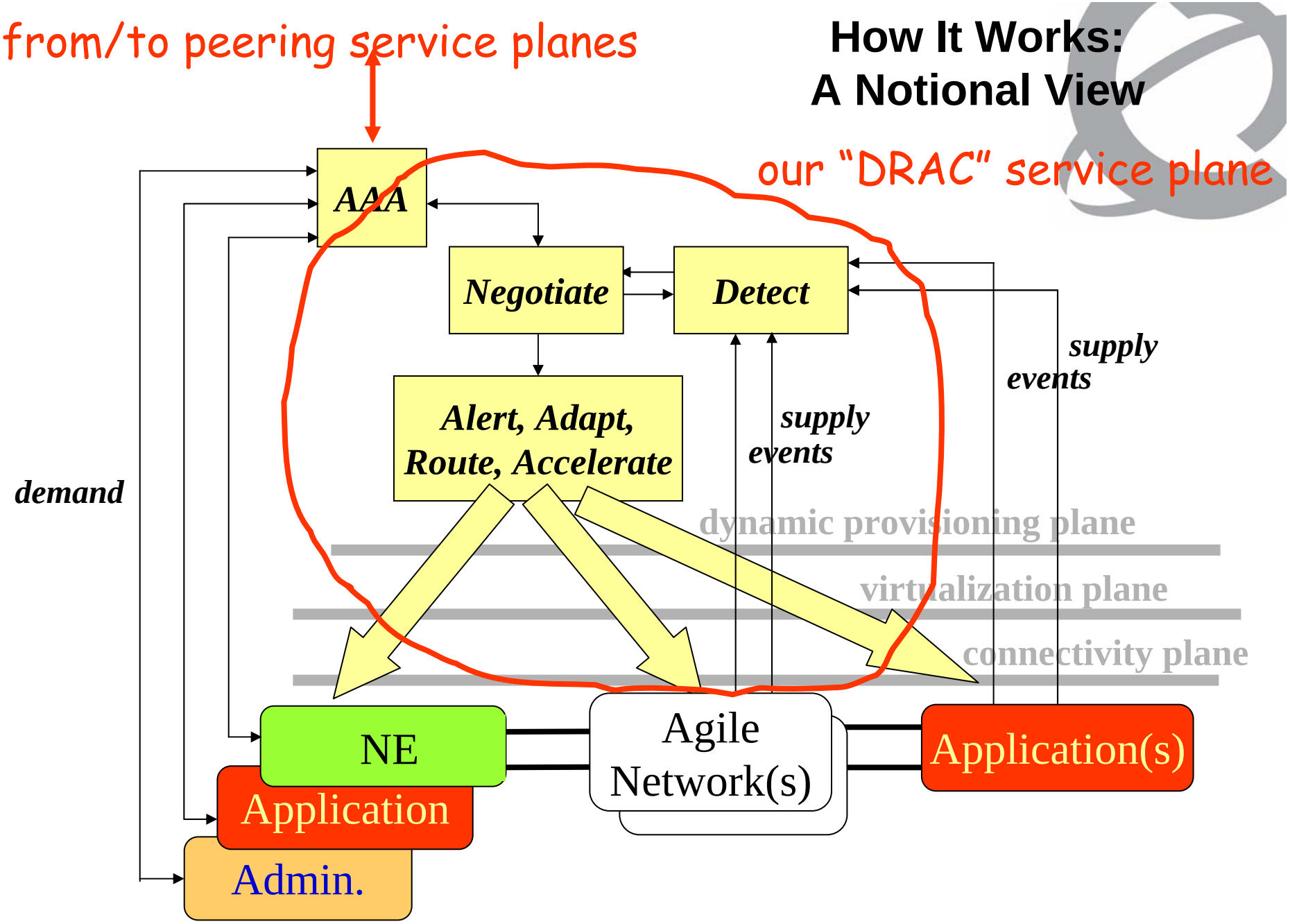
For this, we layer a *service plane* between App and Net

Enabling new degrees of App/Net coupling



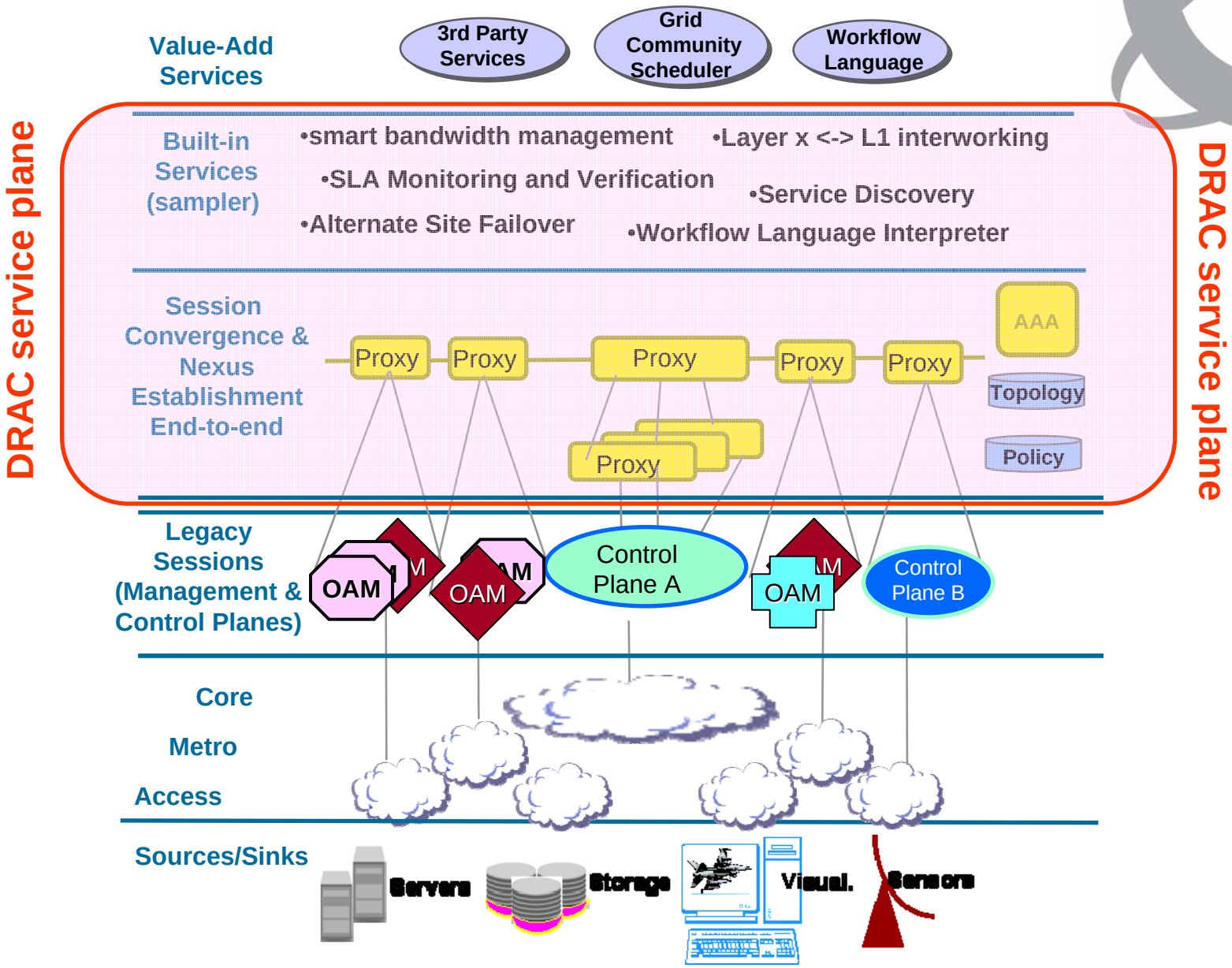
- Hybrid Optical Packet
  - Use ephemeral optical circuits to steer the herd of elephants (few to few)
  - Mice or individual elephants go through packet technologies (many to many)
  - Either application-driven or network-sensed; hands-free in either case
  - Other hybrid networks being explored (e.g., wireless + wireline)
- Application-engaged networks
  - The application makes itself known to the network
  - The network recognizes its footprints (via tokens, deep packet inspection)
  - E.g., storage management applications
- Workflow-engaged networks
  - Through workflow languages, the network is privy to the overall “flight-plan”
  - Failure-handling is cognizant of the same
  - Network services can anticipate the next step, or what-if’s
  - E.g., healthcare workflows over a distributed hospital enterprise
- Cognitive network services
  - Services that can learn from experience, know the ensemble of flight-plans, appreciate missions’ relative merits, and respond optimally to surprise
  - See Dave Clark’s MIT Knowledge Plane and DARPA’s efforts

How It Works: A Notional View



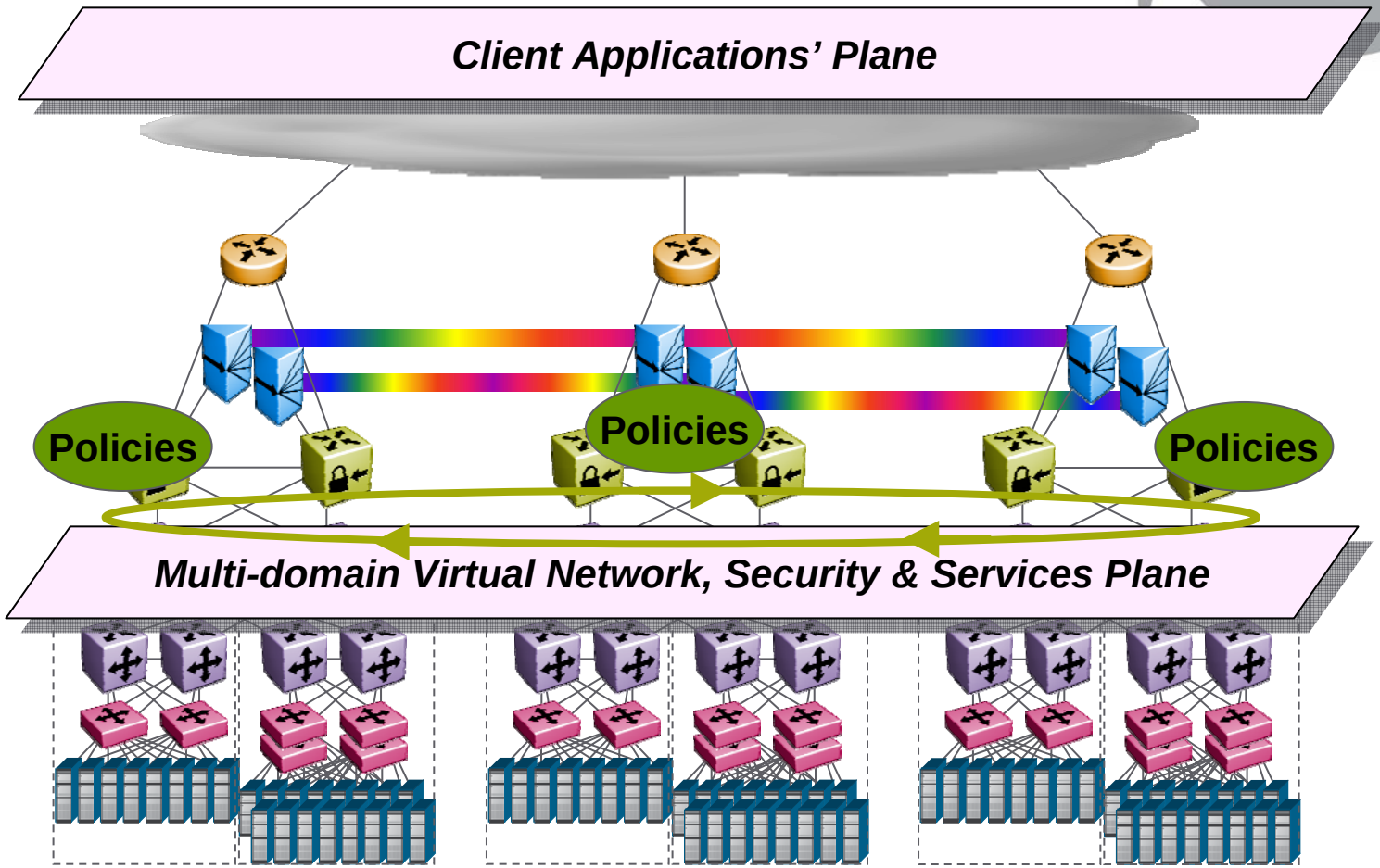
Nortel's Dynamic Resource Allocation Controller (www.nortel.com/drac)

Bird's eye View of the Service Stack

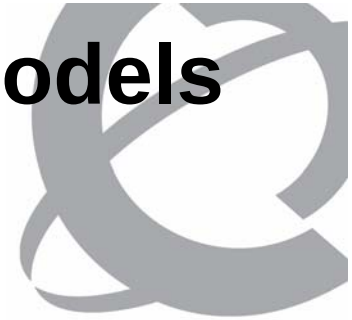


Virtualized Data Centers

the datacomm angle



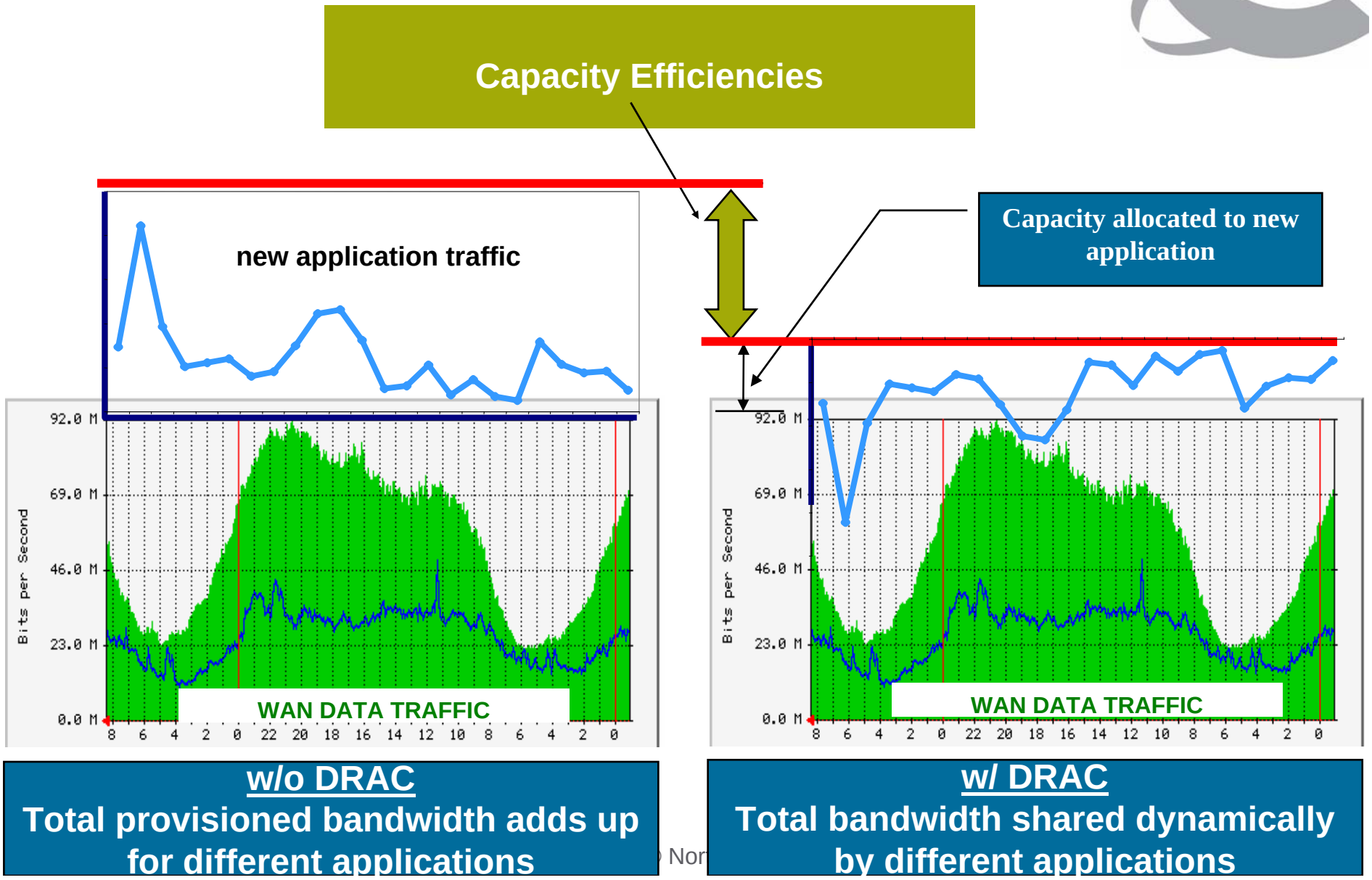
Network as a Service: Programming Models



- conventional
  - Input = {src, dst, service, start time, end time}

- w/ laxity
  - {src, dst, service, estimated duration, must complete by}
  - laxity = deadline – time it would finish if it started at request

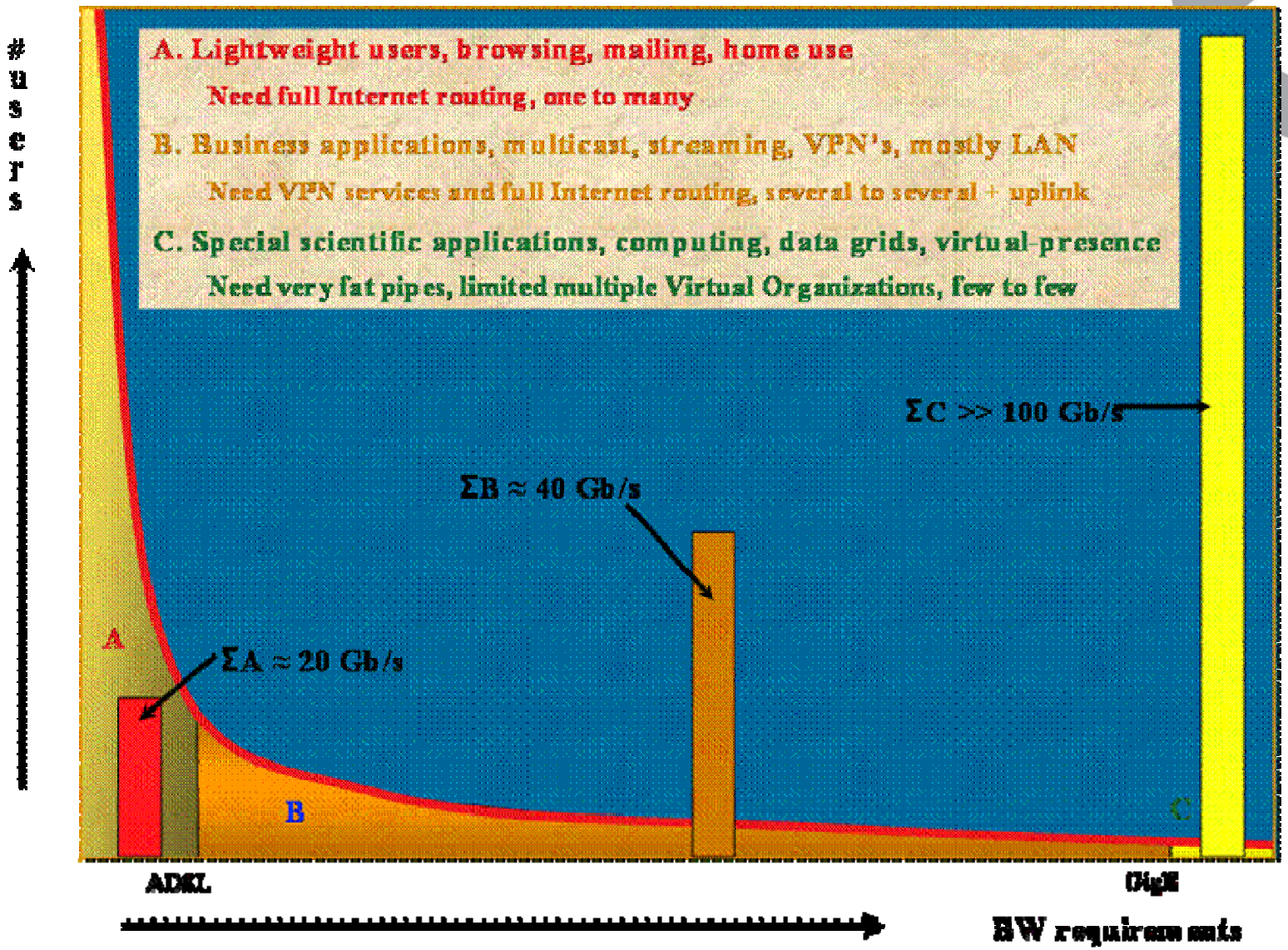
Example #1: Dynamic workflow composition yields capacity efficiencies



w/o DRAC
Total provisioned bandwidth adds up for different applications

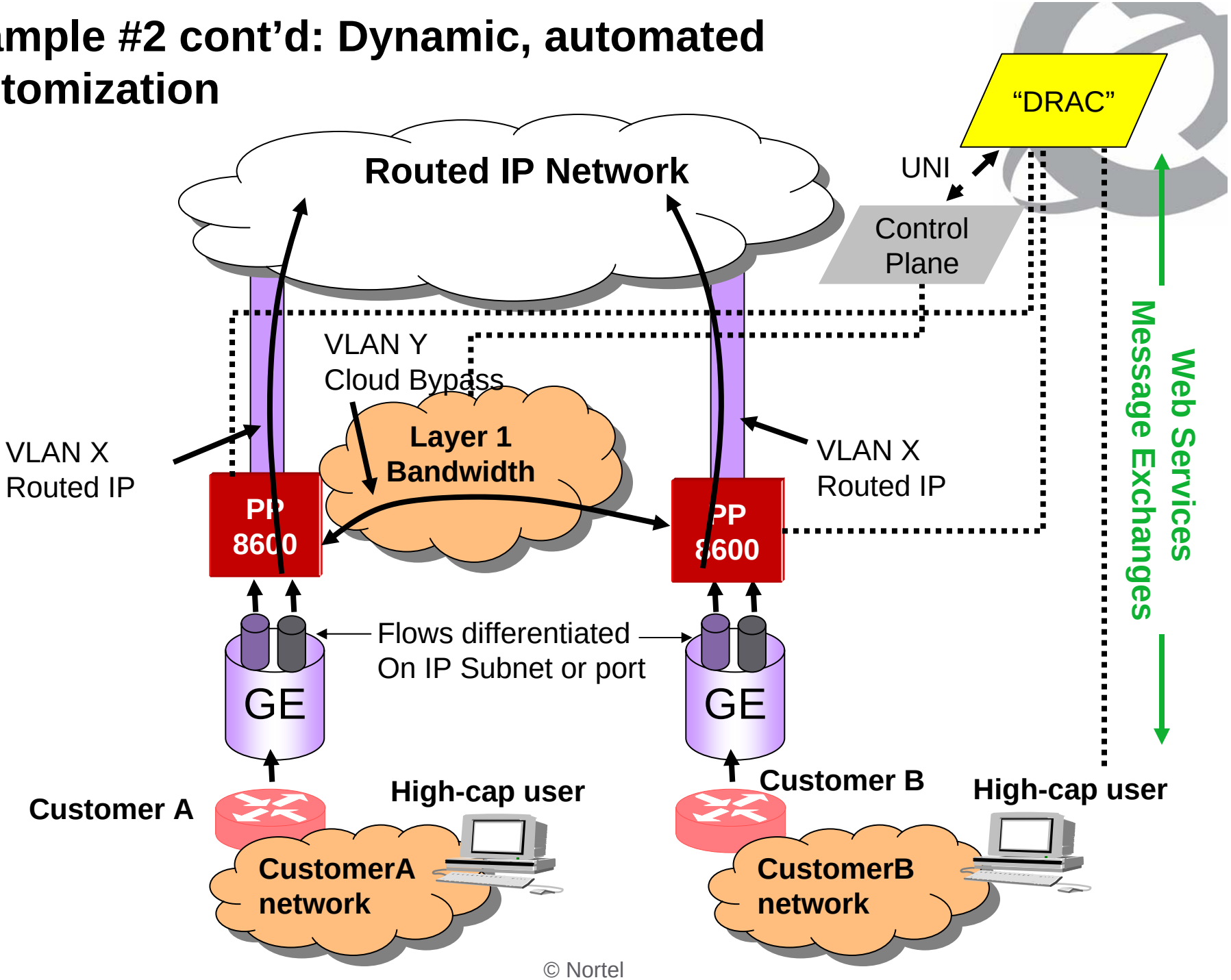
w/ DRAC
Total bandwidth shared dynamically by different applications

Example #2: Dynamic, automated customization



From: C. de Laat, E. Radius, and S. Wallace (2003) "The Rationale of the Current Optical Networking Initiatives," iGrid2002 special issue, *Future Generation Computer Systems*, 19,999–1008

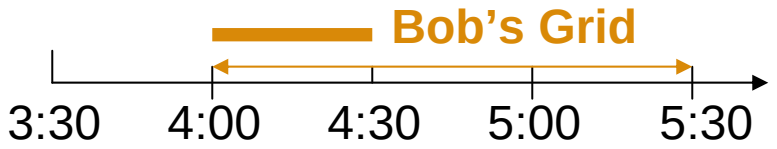
Example #2 cont'd: Dynamic, automated customization



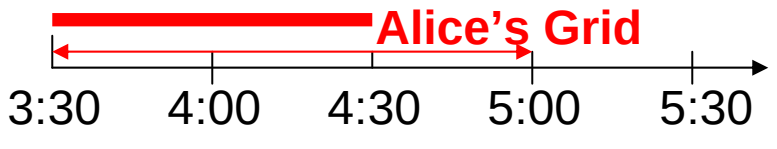
Example #3: Deadline-oriented network allocation w/ laxity

Under-constrained window Use Case

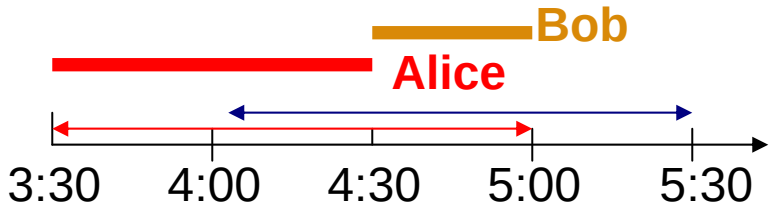
- Request for 1/2 hour between 4:00 and 5:30 on Segment D granted to Bob's Grid at 4:00



- New request from Alice's Grid for same segment for 1 hour between 3:30 and 5:00. Alice's credentials support the request

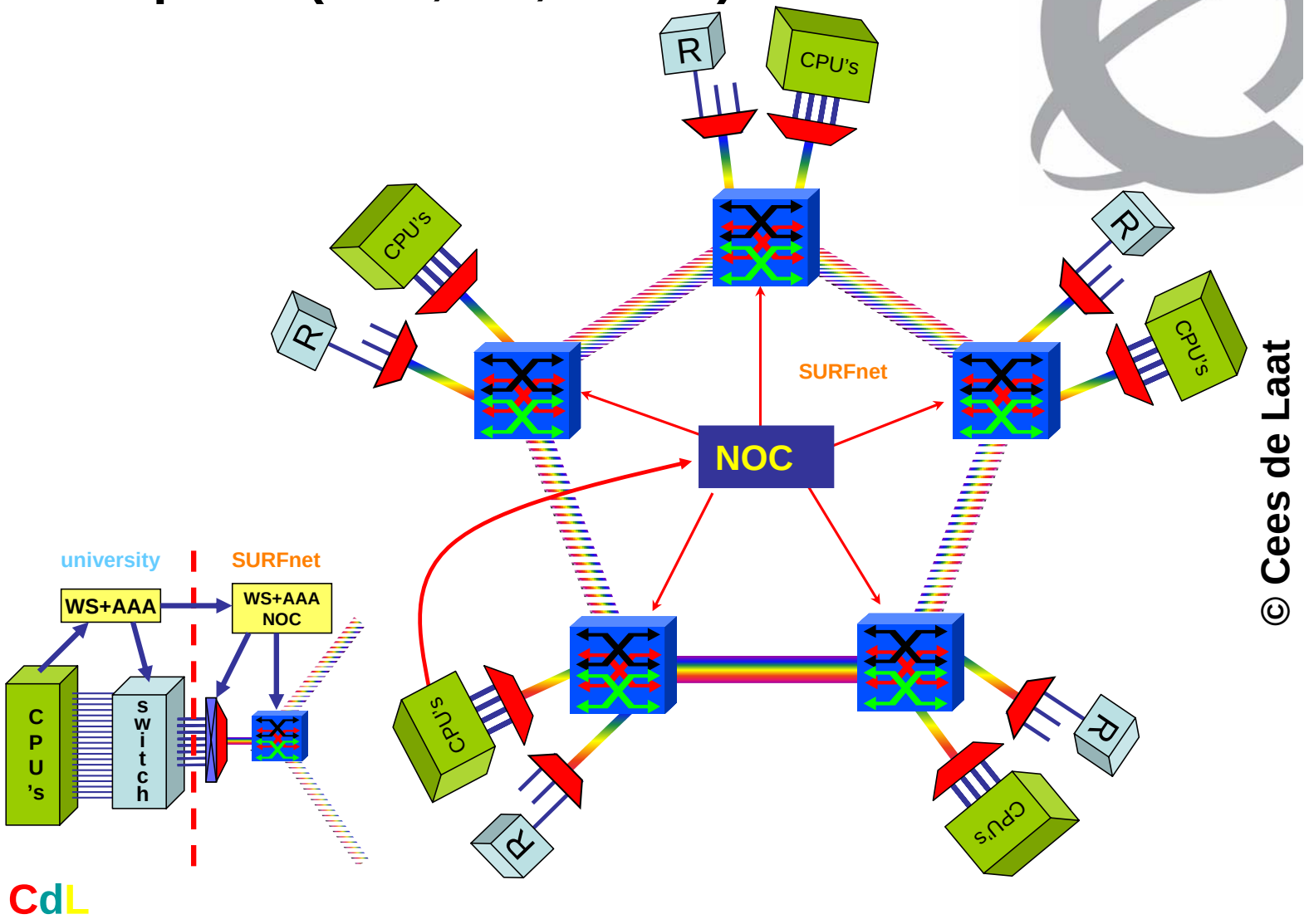


- Reschedule Bob's Grid to 4:30; Alice's Grid stays at 3:30. Everyone is happy.



Network path computed for a time slot; new request comes in; DRAC reschedules former route for a later slot within window

Example #4: Starplane (UvA, VU, Nortel)



© Cees de Laat

> Grid applications drive dynamic allocations in the photonic network infrastructure (a zone of the larger SURFnet6)

Virtualized Computation

Virtualized Networking

mashup \'mashup\ *n*: A mashup is a website or web application that seamlessly combines content and control from more than one source into an integrated experience.

Anatomy of a Mashup

> Mashups are more commonly contextualized within Web 2.0

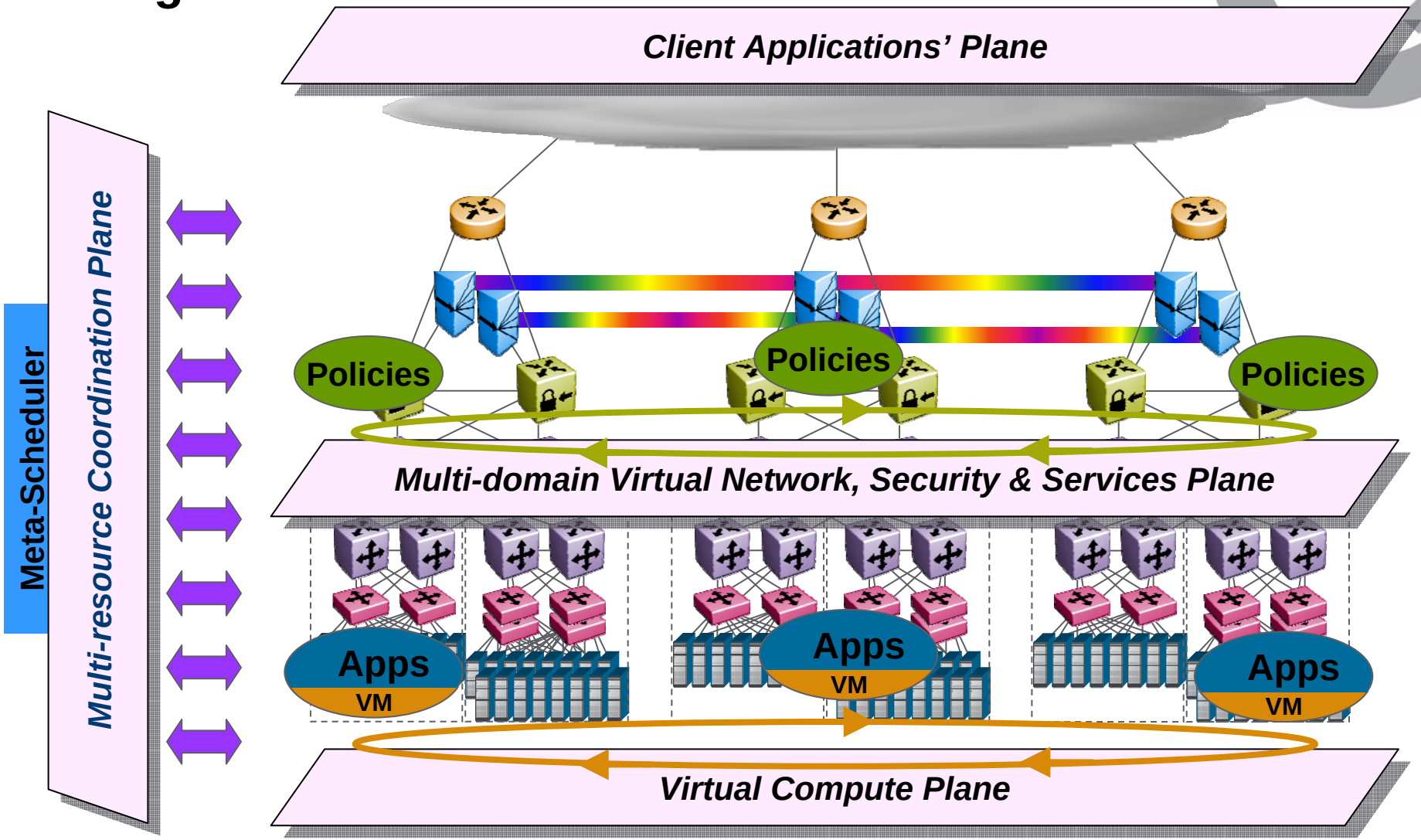
Web 1.0	Web 2.0
DoubleClick	Google AdSense
Ofoto	Flickr
Akamai	BitTorrent
MP3.com	Napster
Britannica Web	Wikipedia
Personal website	Blog
Screen scraping	Web Services
Publishing	Participation
Adapted from Tim O'Reilly	

> Our mashup is atypical in that concerns Infrastructure

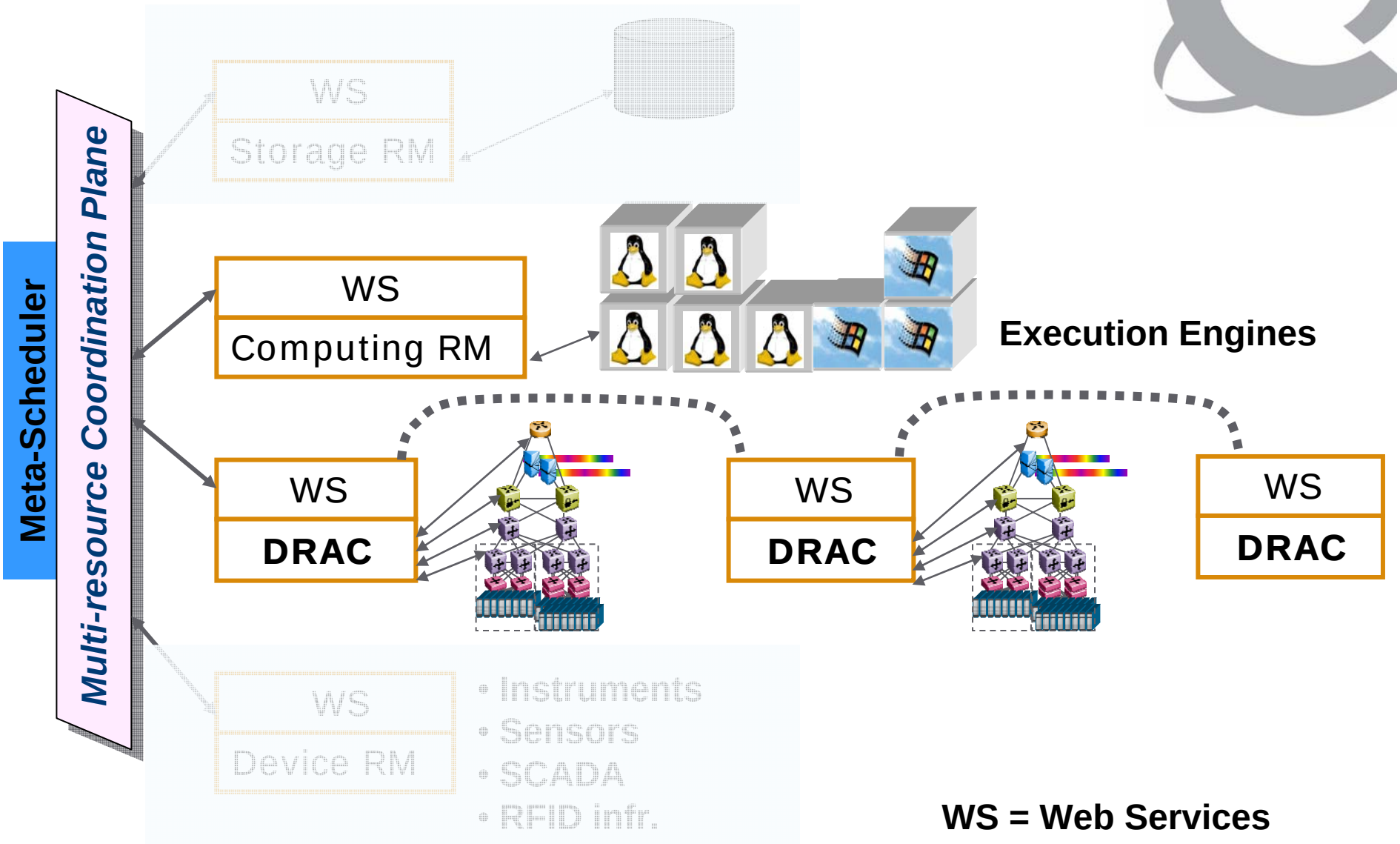
> Though it has many of the common traits

- SOA Architecture, site autonomy, late binding, user control, etc.
- Web Services interfaces

Virtualized Data Centers
w/ integrated resource control



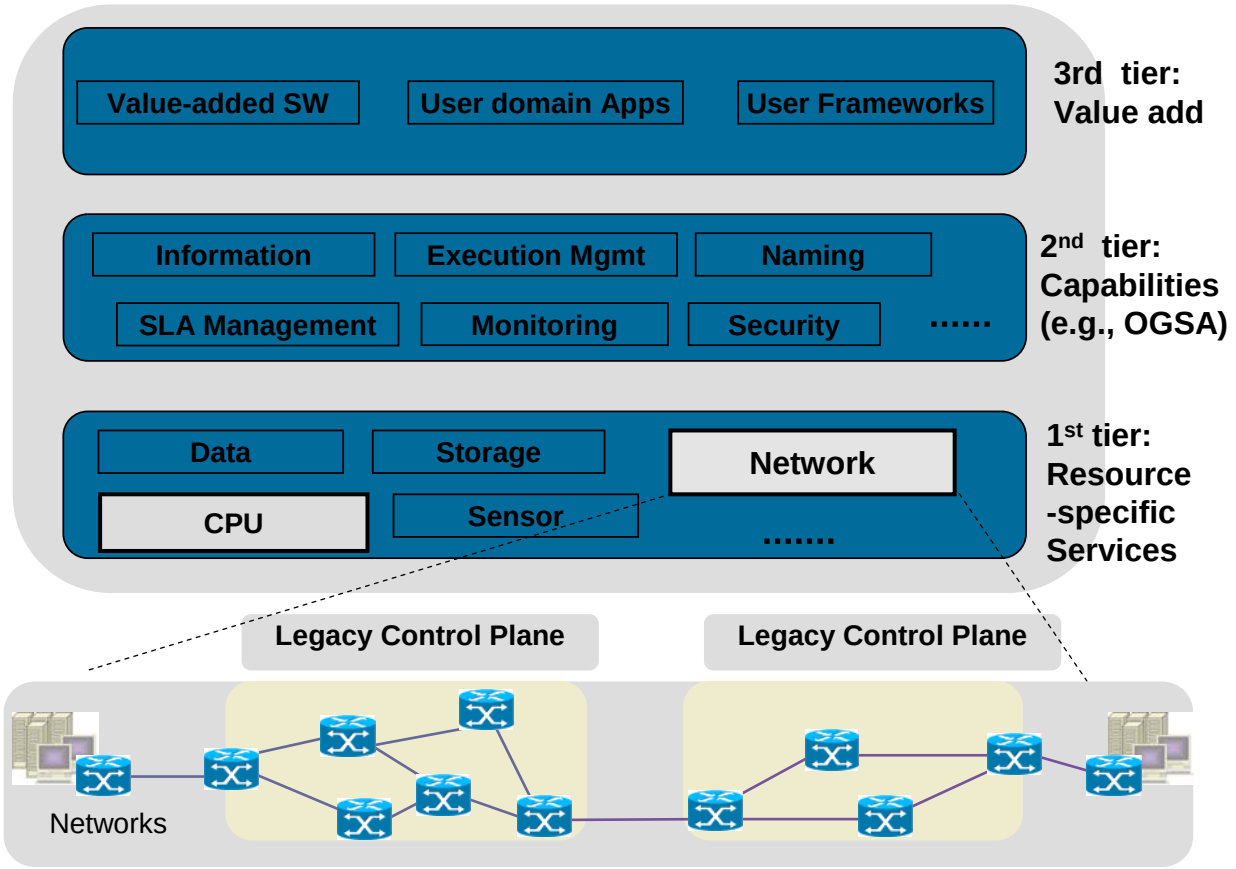
In Focus: Multi-Resource Coordination Plane



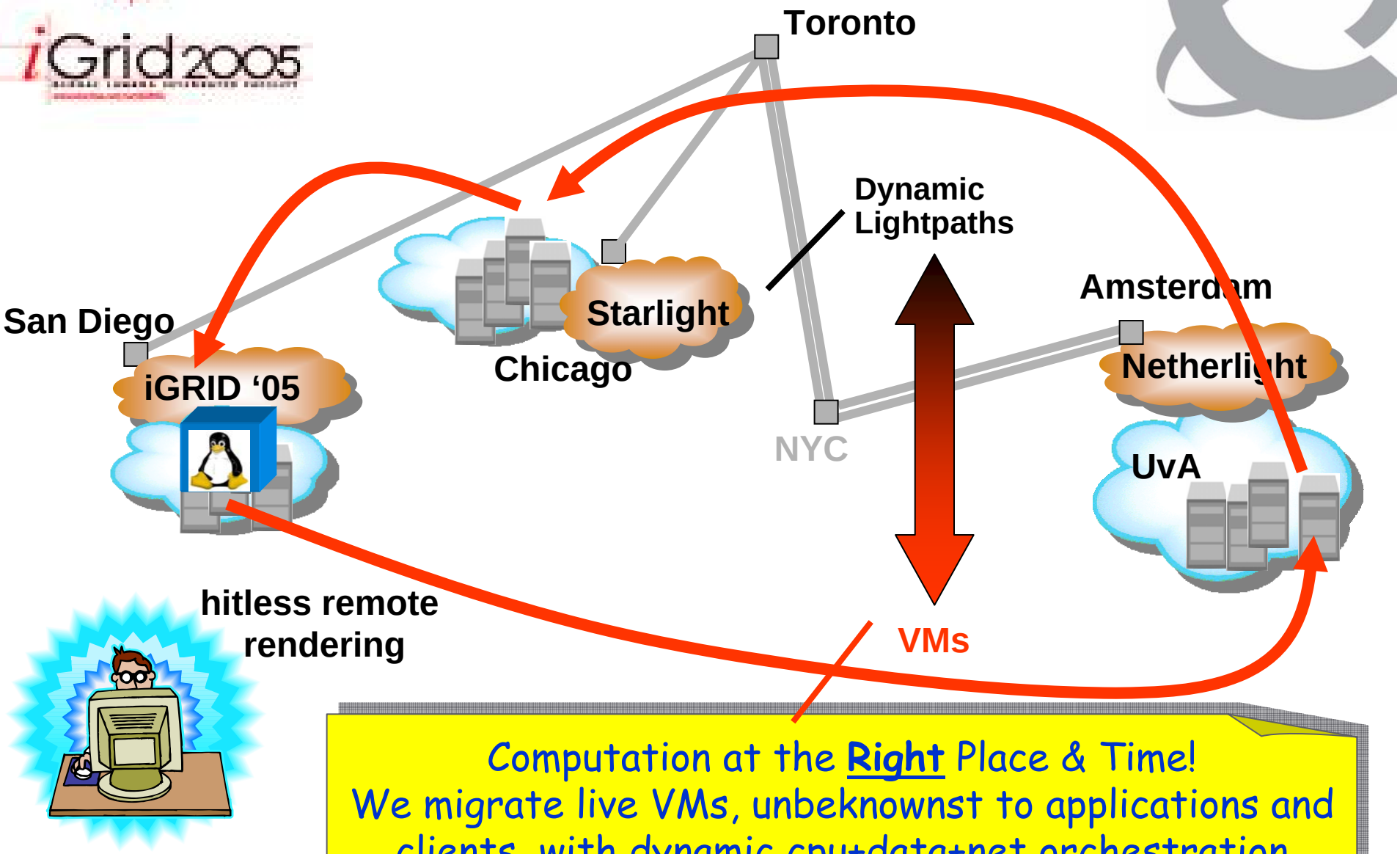
WS = Web Services
RM = Resource Manager

Fit with Grids

- Our mashup results in a resource collective layer for computation and networking
- As such, it's a pillar - not a replacement - for OGSA when Grids are in scope



The "VM Turntable" Demonstrator



Computation at the **Right** Place & Time!
We migrate live VMs, unbeknownst to applications and clients, with dynamic cpu+data+net orchestration

In the Blink of an Eye

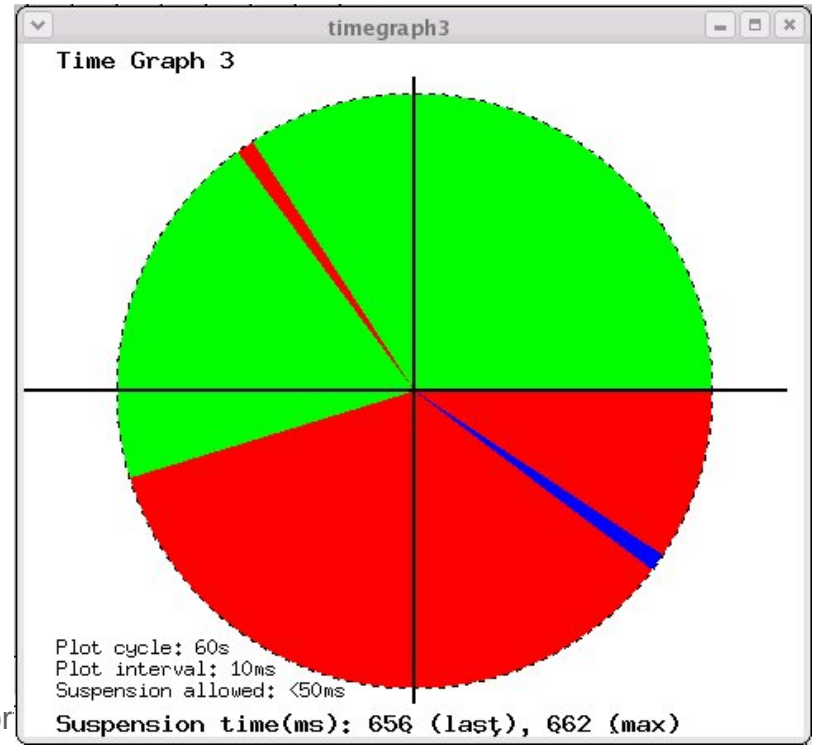
> Virtual Machine teleported over thousand miles

> Seamless to external clients, w/ just a tiny ~1s glitch

> Downtime is limited in spite of high RTTs

- San Diego – Amsterdam, 1GE, RTT = 200 msec, downtime = ~1 sec
- Back to back, 1GE, RTT = 0.2 - 0.5 ms, downtime = ~0.2 sec

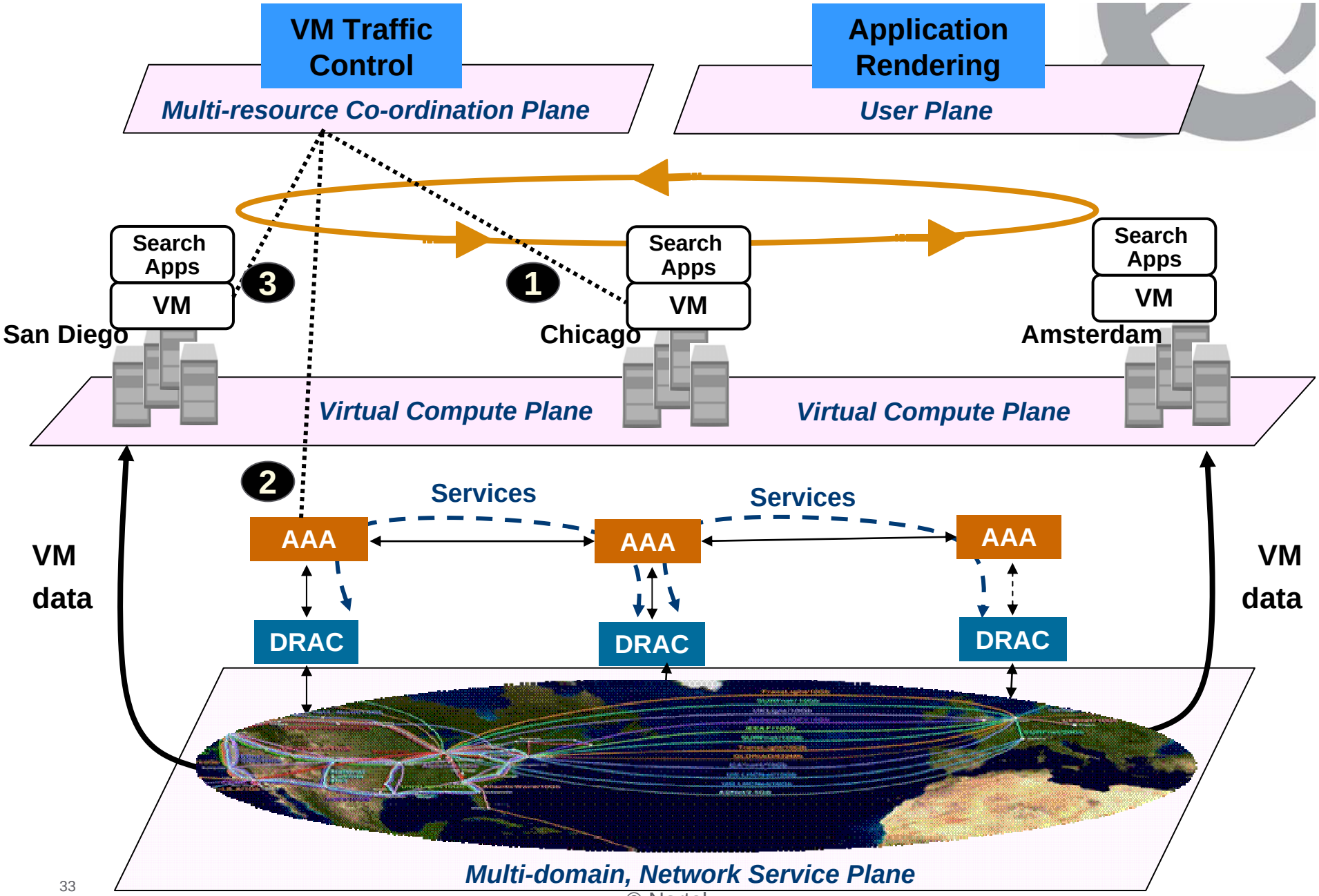
downtime is only ~5x
while RTT is 1,000x !!!



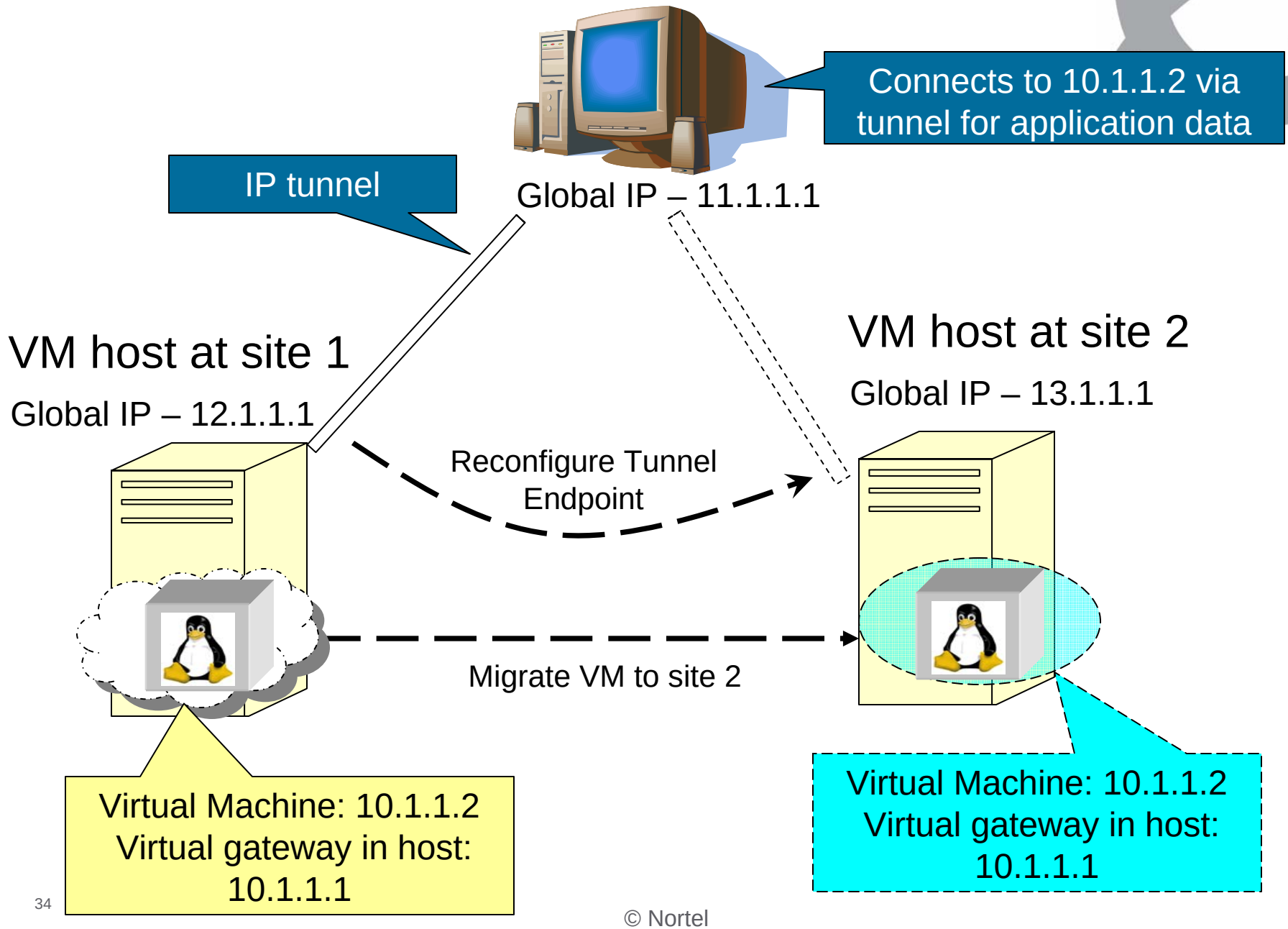
> Lightpath is a virtualized optical link

> Its determinism (not the bw!) is enabling technology

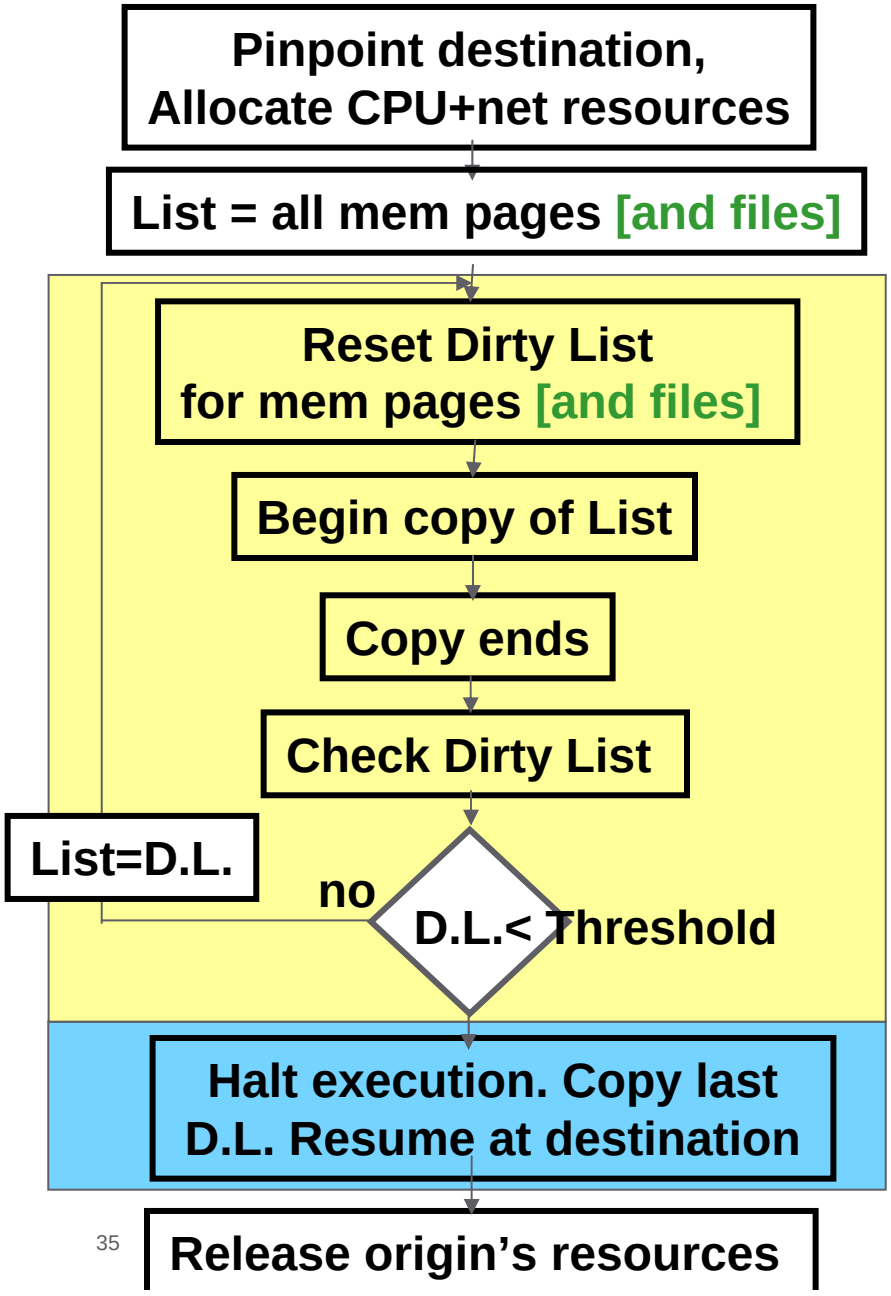
Cross-section: Virtualized User, Compute & Network Planes



Endpoint Migration for Seamless Connectivity



VM Migration Workflow



Packet Routed

w/ Lightpath

no prep cost

AAA, DRAC, path compute, hardware latency

Page dirtying rate vs. network flush time?

AIMD, delay, put convergence at risk!

Net flush time shorter&shorter. Converge in few iterations

May never get here

Short Downtime

What matters for downtime

Data points /1

Table 1: Application-level internal measurement of downtime

Downtime (ms)	Without background image processing	With image processing
Min	279	1877
Max	349	2045
Mean	315	1939

Table 2: Application-level external measurement of downtime

Downtime (ms)	Without background image processing	With image processing
Min	198	1680
Max	280	2120
Mean	230	1940

Table 3: Ping Response Times

Time (ms)	Normal	With Migration
Min	0.030	0.03
Max	0.141	3810
Mean	0.053	0.621

Data points harvested from migrations on the Ams/NYC/Ams loop (RTT = 175 ms)

Data points /2

> Non-live migration 50 – 100 greater than live migration

> Cold lightpath establishment over multiple domains = 40-60s
  - Migration’s downtime unaffected
  - Un-optimized code still

Lessons learned w/ VM Turntable

> The “lightpath” service yielded the required predictable performances. Quite good even when the lightpath was carried on a lightly-loaded layer 2 network

> Iterative pre-copy of memory pages while applications are running avoids negative impacts of TCP
  - Specialized L4 bit blasters wouldn’t yield sensible gains

> Service discovery must factor in attributes like Jumbo frames
  - Had to fix Xen to properly handle 8k frames

> The measurement of actual downtime, quite an elusive task

Further Thoughts #1 — Impactful Paradigm

> Growing interest around moving computation towards jumbo directories
  - Example: www.alexa.com
  - Emerging e-Utilities

> VM is a powerful unit of service
  - Configure LVM-like support for state that was committed to storage

> Especially when managed via a Web Services wrapper
  - Which abstracts lifecycle operations on a VM
  - And idiosyncrasies of VM implementations

Further Thoughts #2 — Token-based security

> To secure lightpaths, we used Leon Gommans' (UvA) token work and University of Amsterdam's AAA

> The token is a crypto-strong concise capability

- It can be passed out-of-band or in-band
- It opens a "padlock" governing access to the lightpath

> This approach applies to resources other than lightpaths

- i.e., the converged "currency" for cpu + data + network allocation

> Technology push

- An in-band token can easily be validated at 10 Gb/s and beyond

> And market pull

- Providers just love pre-paid semantics !

Further Thoughts #3 — RDMA

> Today, migration comes with an I/O tax

> A better way is to drive lightpaths straight into memory

- main processor no longer has to marshall/unmarshall
- no interrupts
- zero copy semantics
- NIC++ performs these tasks

> Enter RDMA

- Infiniband™ is an example of fabric which realizes RDMA natively
- A HCA is endowed with plenty of silicon resources
- Well suited to copy OS pages during migration

Conclusions

> Virtualized computation
  - Isolation, consolidation, migration

> Virtualized networking
  - Dynamic resource allocation for high-touch services

} Mashup!

> The VM-Turntable demonstrator shows cpu + network multi-resource coordination in action

> We can steer computation towards data, even at planet scale, unbeknownst to applications and users
  - Live migration in as low as 1 second
  - Pipelining aptly contains downtime despite high RTTs

Support this momentous journey

Old World

Static

Silo

Physical

Manual

Application

New World

Dynamic

Shared

Virtual

Automated

Service