QoS Granularity: resource sharing, virtualization, and application perspective

Jim Kurose
Department of Computer Science
University of Massachusetts
Amherst MA USA

Overview

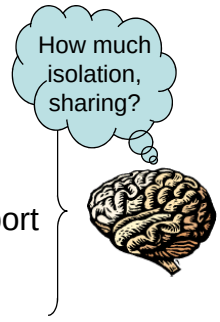
- QOS retrospective
- what have we learned?
- meeting “end-user” needs: sensor net scenario
  - embedding end-user needs
  - virtues of virtualization
- conclusions

Overview

- QOS retrospective
- what have we learned?
- meeting “end-user” needs: sensor net scenario
  - embedding end-user needs
  - virtues of virtualization
- conclusions

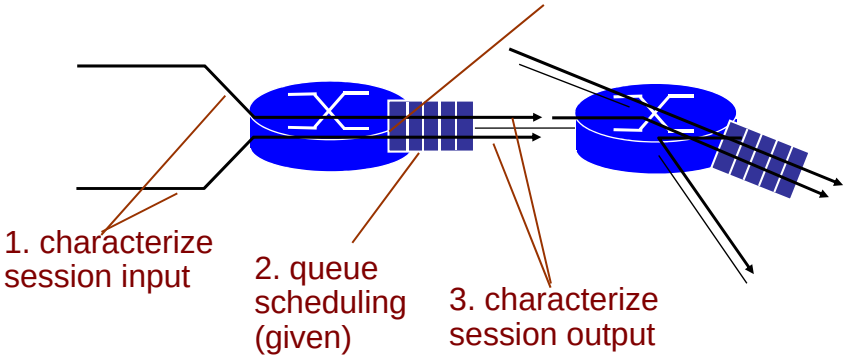
Overview

- QOS retrospective
  - resource reservation
  - service differentiation
  - over-provisioning, application-level support
  - isolation: VPNs
- what have we learned?
- meeting “end-user” needs: sensor net scenario
  - embedding end-user needs
  - virtues of virtualization
- conclusions



Back in the day: resource reservation

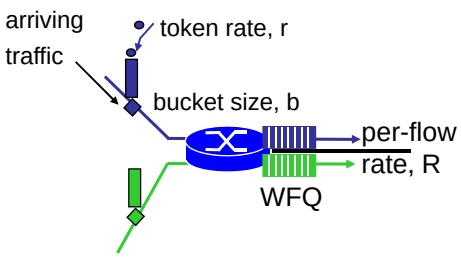
(early-mid 1990's)



IETF Intserv QoS: [rfc2211, rfc 2212]

Guaranteed service:

- worst case traffic arrival: leaky-bucket-policed source
- simple provable) *bound* on delay [Parekh 1992, Cruz 1988]
  - min-plus algebra (Chang98, Cruz99, LeBoudec 2001)
  - bounds on delay distribution (Kurose92, Sidi93, Kesidis00, Chang01, .. Giacomomazzi 2008)



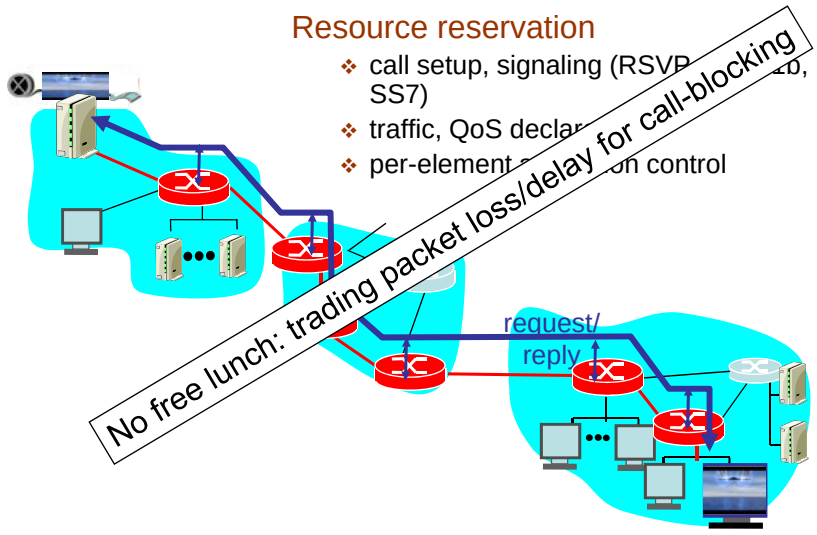
ATM CBR Service:

- per-virtual circuit end-end *constant bit rate* (jitter-controlled) service
  - related circuit emulation service (CES), e.g., for T1/E1 TDM circuits
  - available in products (e.g., Cisco)

Effective bandwidths:

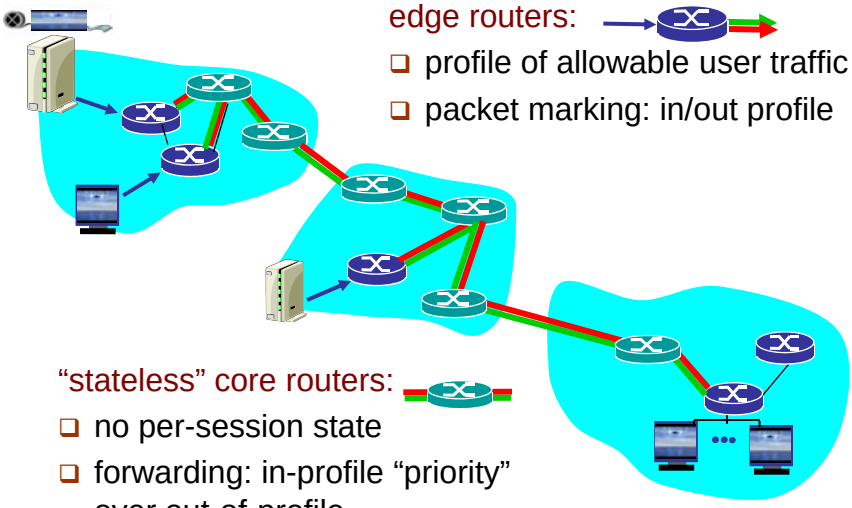
- characterization of per-source “bandwidth needs”: multiplexer meets performance guarantee if sum of effective bandwidths of admitted sessions < 1.
- Guerin(1991), Kelly (1991), Chang (1993), Mitra (1993), Li (2007) ….

Call admission



Next “big thing”: differentiated service

(mid-late 1990’s)



edge routers:

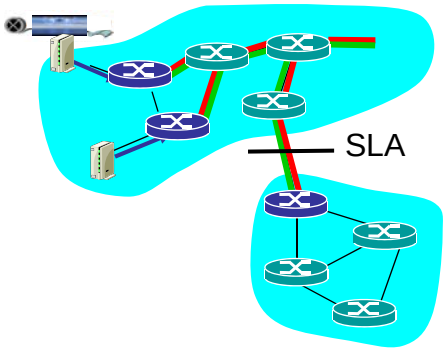
- profile of allowable user traffic
- packet marking: in/out profile

“stateless” core routers:

- no per-session state
- forwarding: in-profile “priority” over out-of-profile

Differentiated Service: challenges

- over-provisioned bandwidth for in-profile traffic : how much?
- from aggregate to per-flow performance
- end-to-end issues
- MPLS: alternate approach to service differentiation
  - just let us (NOC) do it



Next big thing: overprovisioning, application infrastructure (early 2000's – today)

- throw resources at problem: enough bandwidth to "ensure" no contention for resources
  - inefficient use of resources, but …
  - research question: how much over-provisioning is enough (e.g., Guerin 05, Roberts …)
- enhance service using sophisticated application-level infrastructure (caches, relays, peers)
  - Skype supports 8M VoIP callers
  - PPlive: IP TV
  - video via BitTorrent
  - Push2Peer nano data centers

Meanwhile, in industry (1990's to today)

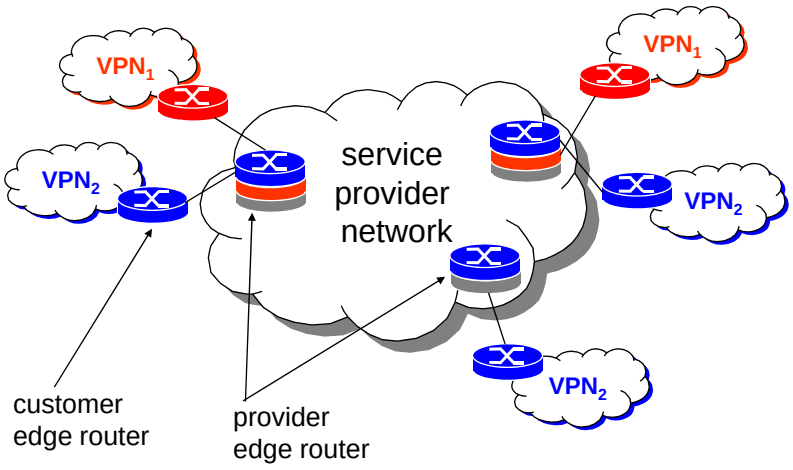
- *Q:* What do our customers want?
- *A:* Their own network!

VLANS, VPNs

Networks perceived as being private networks by customers using them, but built over shared infrastructure owned by service provider (SP)

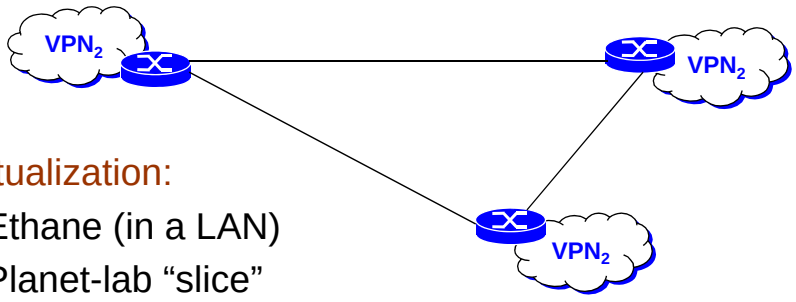
- SP infrastructure: backbone, provider edge devices
- customer: edge devices (communicating over shared backbone)

VPN reference architecture



VPN: logical view

- logical isolation from other VPNs
  - multiplexing/performance issues within VPN

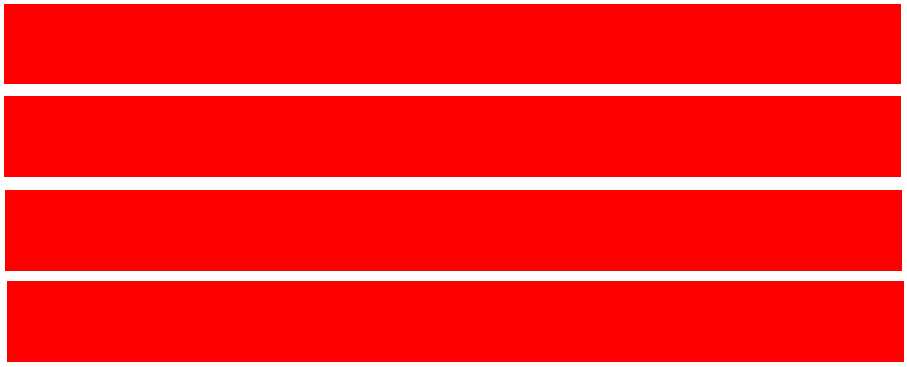


Virtualization:

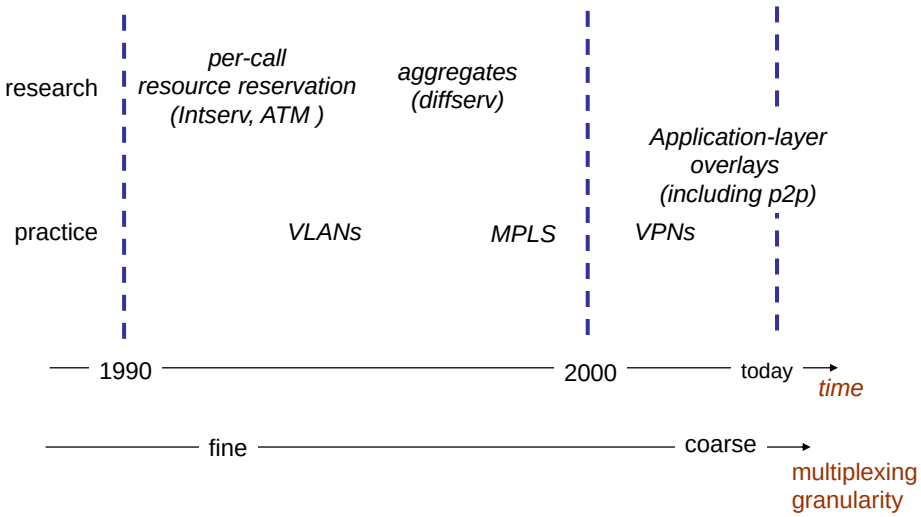
- Ethane (in a LAN)
- Planet-lab “slice”
- VINI (with lots of programmability)

QoS retrospective: summary

Unit of allocation | guarantee | deployment | complexity



QoS retrospective: summary

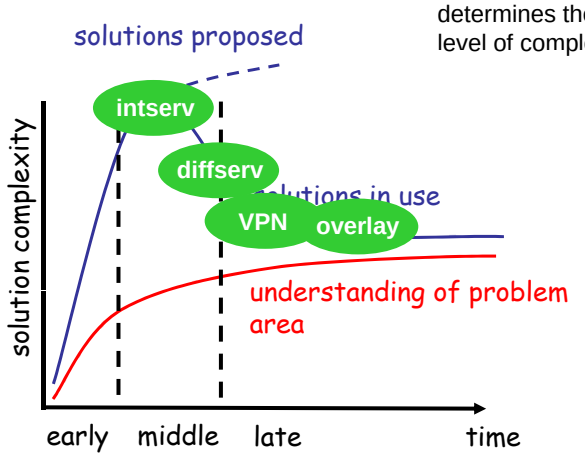


Overview

- QOS retrospective
- what have we learned?
- meeting "end-user" needs: sensor net scenario
  - embedding end-user needs
  - virtues of virtualization
- conclusions

The right level of complexity

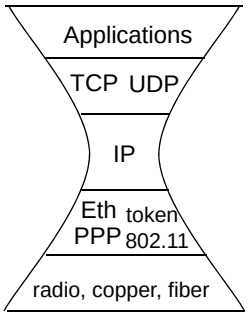
Q: What process determines the "right" level of complexity?



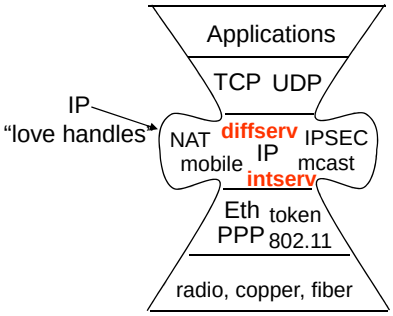
[adapted from Hluchyj 2001]

Middle-aged Internet: losing the hourglass?

middle age: a narrowing mind, a widening waist



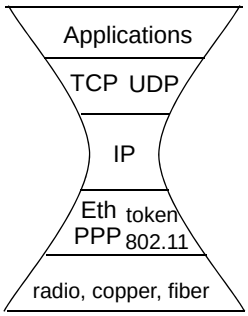
Youthful, IP “hourglass”



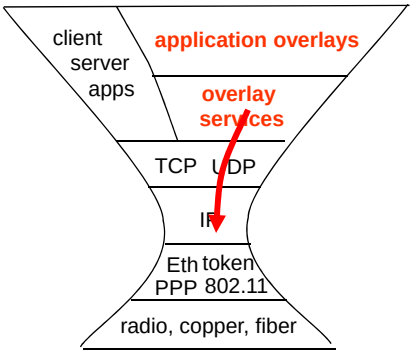
Internet at middle-age: lovehandles? ?

Middle-aged Internet: *keeping* the hourglass!

middle age: a expanding mind, a slim waist



IP “hourglass”



Overview

- QOS retrospective
- what have we learned?
- meeting “end-user” needs: sensor net scenario
  - embedding end-user needs
  - virtues of virtualization
- conclusions

QoS meets sensor networking

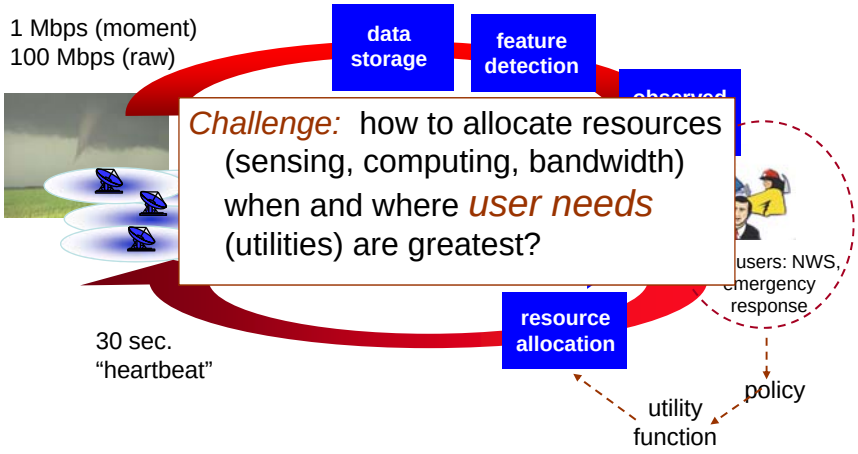
Scenario:

- multiple users access remote sensors
- sense-and-response: *actuated* sensors, *controlled* by user requirements
  - competing, conflicting user resource demands (sensor targeting, computation, bandwidth)

Challenge:

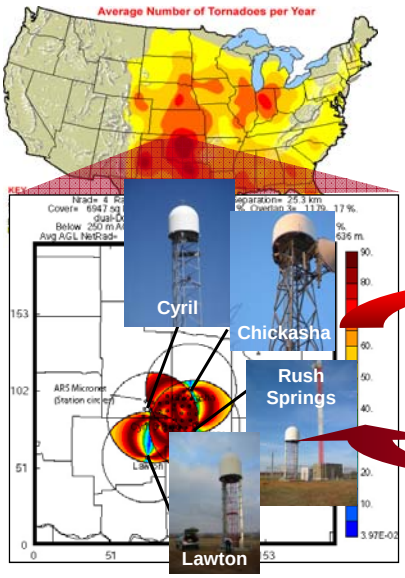
- how to allocate resources when and where user needs (utilities) are greatest?
  - multi-resource, end-end QoS

Sense-and-response control loop



CASA: Collaborative Adaptive Sensing of the Atmosphere

Oklahoma 4-Node Test Bed

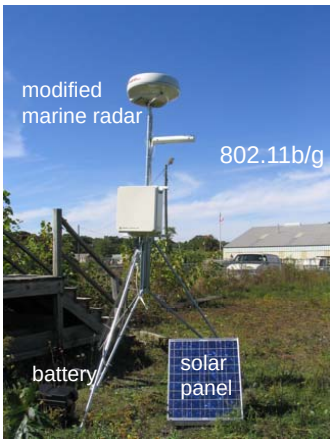


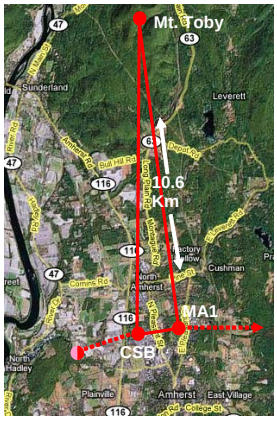
End users: national weather service, emergency response managers, researchers

Norman OK (NOC)

Off-the-Grid Test Bed

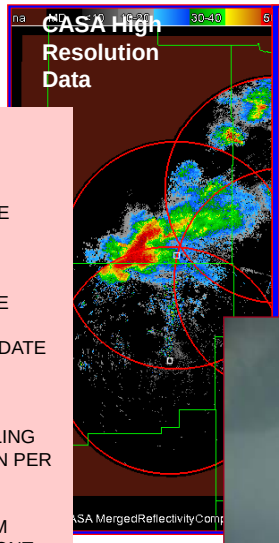
- no reliance on infrastructure
- solar/battery-operated nodes
- multi-antenna multi-hop 802.11 directional antenna
- GENI

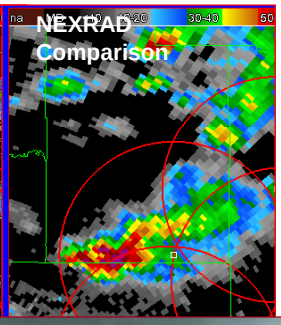




Topology of the MA OTG testbed

2007 storm season in Oklahoma





FLUS74 KOUN 102343
AWUOUN

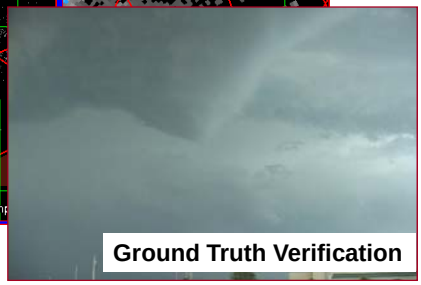
AREA WEATHER UPDATE
NATIONAL WEATHER SERVICE
NORMAN OK
742 PM CDT TUE APR 10 2007

..WARNING DECISION UPDATE

THIS WARNING DECISION UPDATE CONCERNS COMANCHE AND GRADY COUNTIES.

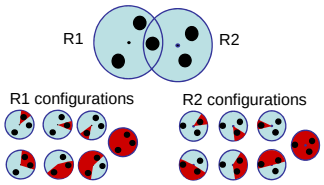
MESOCYCLONE NEAR STERLING CONTINUES TO STRENGTHEN PER TWO RADAR VIEWS. ***CASA NETWORK ALSO SHOWING PRONOUNCED HOOK.*** STORM WILL ENCOUNTER WARM FRONT...

BURKE



Embedding user preferences: utility functions

Where to point?



Find ***configuration*** that optimizes utility at time step k:

$$J = \max_{configurations, C} \sum_{tasks, t} U(t,k) Q(t,C)$$

Utility – **"how important"** is task *t* to the users at time k?

$$U(t,k) = \sum_{groups, g} w_g U_g(t,k)$$

Quality – **"how good"** is scanning configuration *C (distance, coverage, # radars) for task t?*

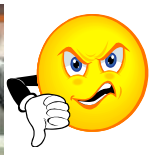
How to define "how important": $U_g(t,k)$

- user values for detected weather features

Event	Location	Prior Information available	NWS utility Wt=0.4	EM utility Wt=0.3	Researcher utility Wt=0.2	Vieux utility Wt=0.1
TVS detection	AOP	0	5	5	5	1
		1	4	5	5	1
	Remote	0	5	1	5	1
		1	4	1	5	1
Mesocyclone	AOP	0	4	4	4	1
		1	3	4	4	1
	Remote	0	4	1	4	1
		1	3	1	4	1
Storm cell	AOP	0	4	4	4	4

How to define "how important": $U_g(t,k)$

- "naturally": group-sensitive utility for each feature (tornado, wind shear, hail core) scanned
- … and the survey says…..



User feedback:

- NWS: want "mental movie" scanning "areas of interest" at regular intervals
- need context: scan areas around features (storm cell)
- want to joystick system (want their own network)

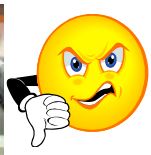
User Utility Rules (revised)

- *interval-based preferences:* "do X every Y time units"
- utility considers both objects, time

Rules	Rule trigger	Sector Selection	Elevations	# radars	Contig.	Sampling interval
NWS						
N1	time	360	lowest	1	Yes	1 / min
N2	storm	task size	[illegible]	1	Yes	1 / 2.5 min
EMs						
E1	time	360		1	Yes	1 / min
E2	reflectivity over AOI	task size		1	Yes	1 / min
E3	velocity over AOI	task size	lowest	2+	Yes	1/ 2.5 min

How to define "how important": $U_g(t,k)$

- "naturally": group-sensitive utility for each feature (tornado, wind shear, hail core) scanned
- … and the survey says…..



User feedback:

- NWS: want "mental movie" scanning "areas of interest" at regular intervals
- need context: scan areas around features (storm cell)
- want to joystick system (want their own network)

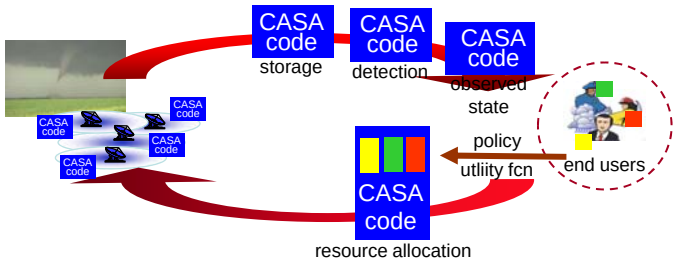
- want to joystick system (want their own network)

Virtues of Virtualization

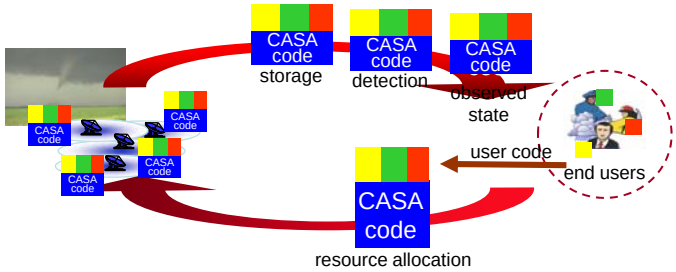
- *virtualization* of computing, communication, *and sensing* resources
- *each* user:
  - sees "standalone" sensor network
  - can modify, download, execute, experiment with own code
  - implements user-specific service outside (architecturally above) infrastructure provider

Virtualization: enabling end user

instead of this….

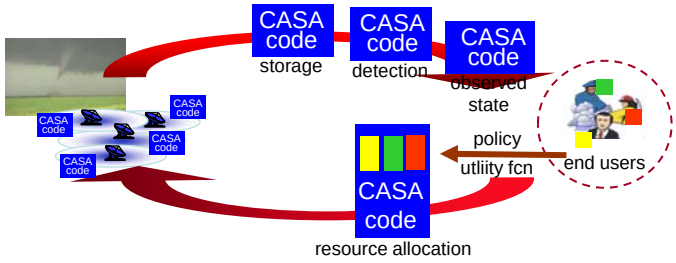


….this system view

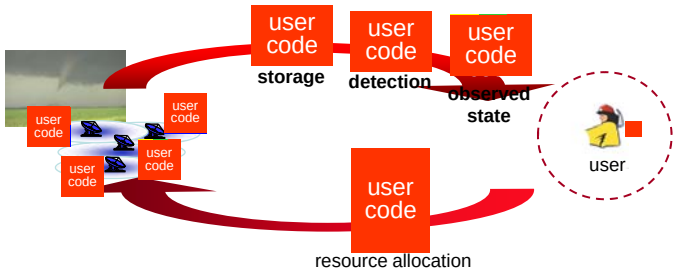


Virtualization: enabling end user

instead of this….



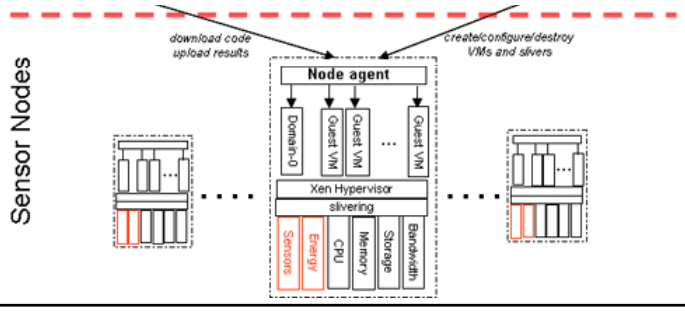
logical user-view: dedicated system!



Challenges

- virtualizing sensing resources:
  - *sharing:* sensed data from one user usable by another (unlike bandwidth, computing)
  - *admission control:* mediating among different users with different priorities
    - what to do when users' request partially satisfiable? (negotiate?)
    - time-vary allocation of resources?
    - priorities among users (policy)

Virtualization Architecture



Overview

- QOS retrospective
- what have we learned?
- meeting “end-user” needs: sensor net scenario
  - embedding end-user needs
  - virtues of virtualization
- conclusions

The *really* big picture

- importance of user requirements

“It’s the end-user, stupid”

“It’s the application, stupid”

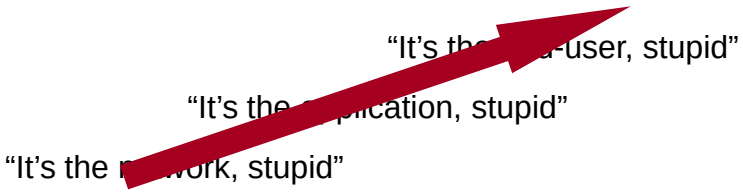
“It’s the network, stupid”

of course, not everyone agrees ….

Verizon product, 2008

The *really* big picture

- importance of user requirements



- architecture (as opposed to stovepipe) for embedding user requirements into network?
  - sensor networks
  - content distribution
  - special-purpose overlays

Fine

grazie mille!

?? || /* */