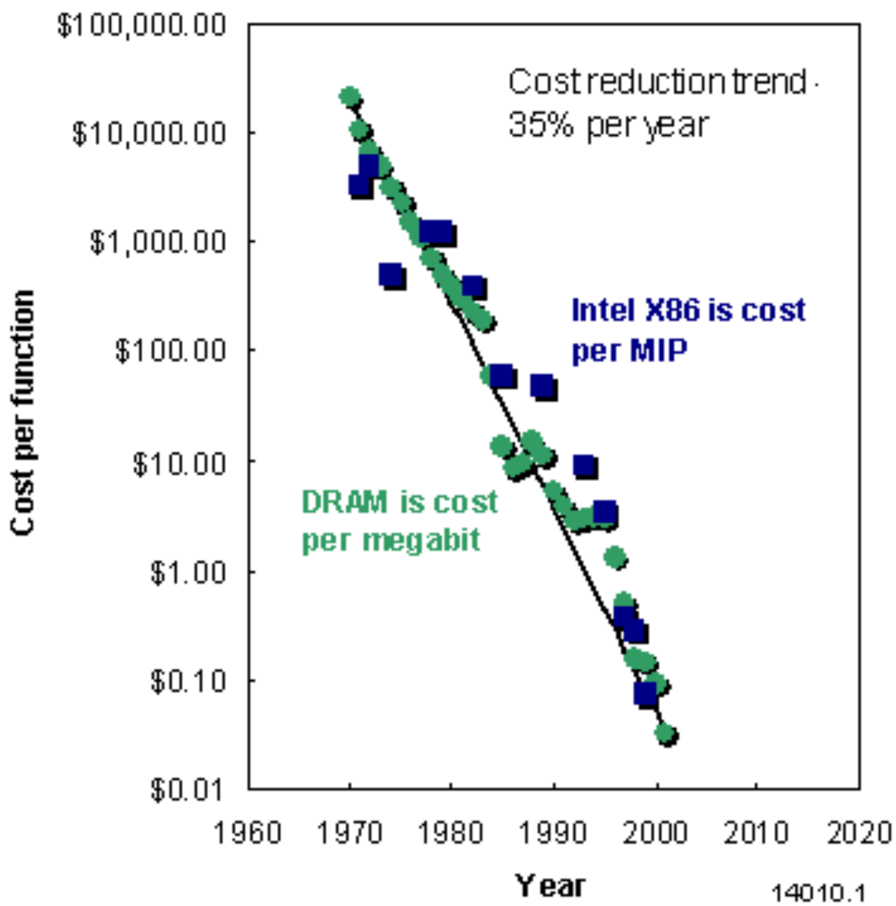
Towards Tetherless computing

S. Keshav
Ensim

Outline

- Technology trends
- Vision for tetherless computing
- Research areas
  - Virtualization
  - Internet Data Center topology
  - Fast, secure roaming
- Conclusions

1. Computing costs are plummeting



From www.icknowledge.com

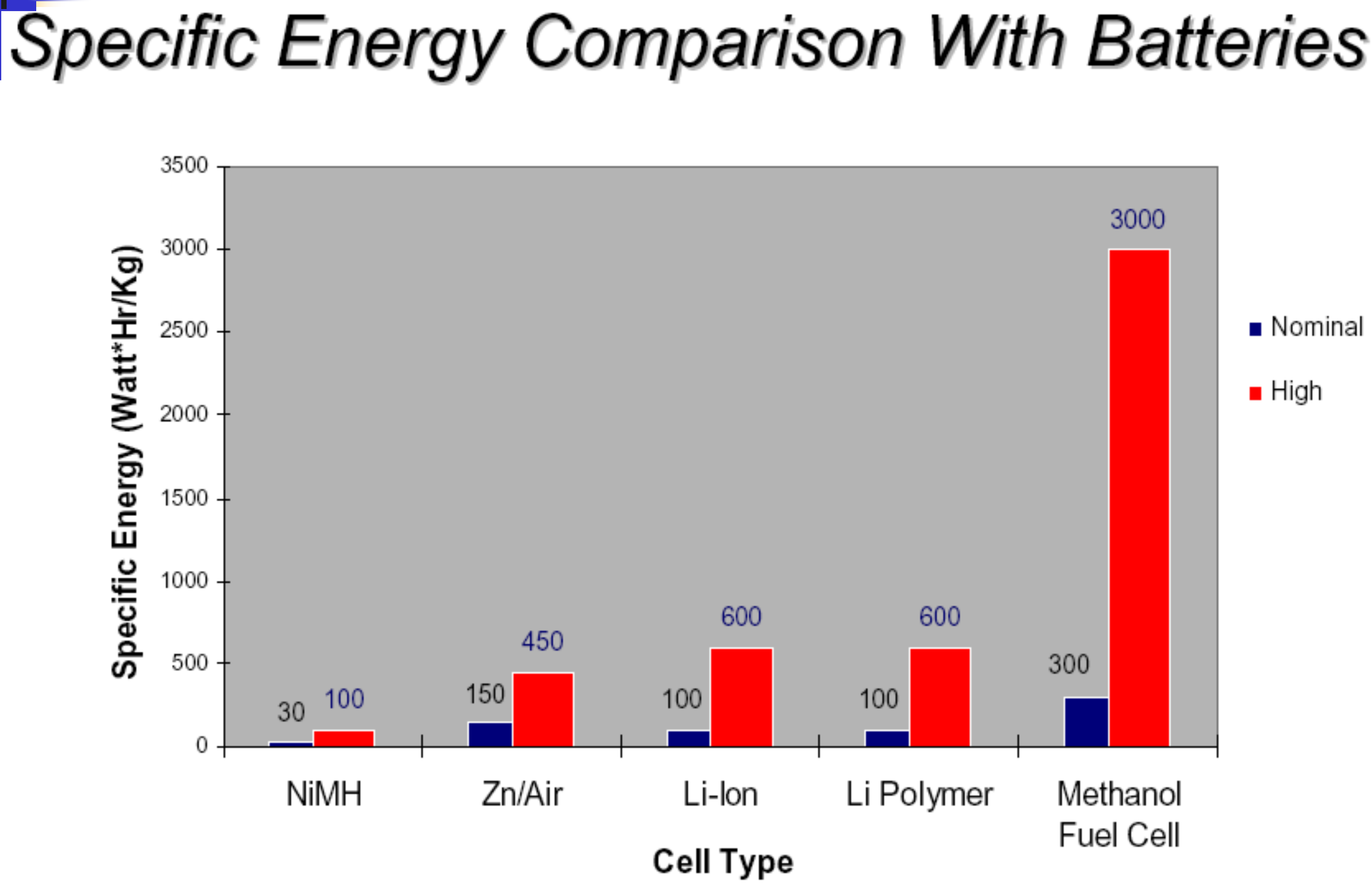
Processor costs have come down by six orders of magnitude in three decades

CMOS allows on-chip logic, memory, imaging and RF components

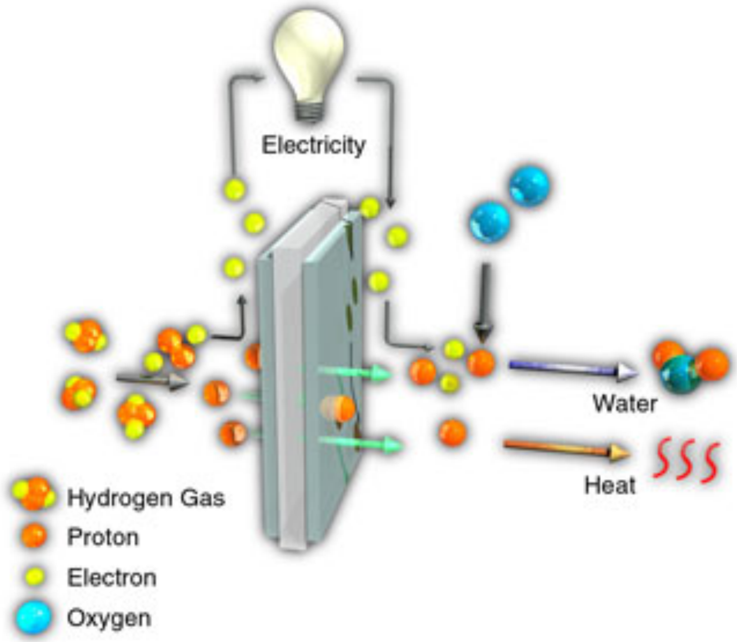
Devices will merge computing, audio, and video

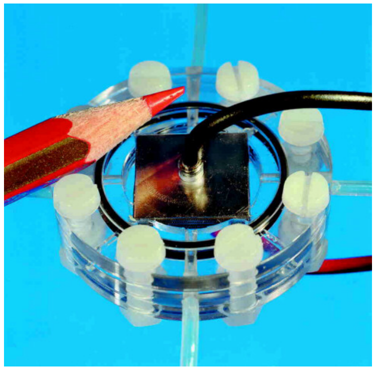
- Processor
- RAM
- Flash memory
- Cell phone modem
- Still camera
- Video camera
- MP3 player

2. Batteries are lasting longer

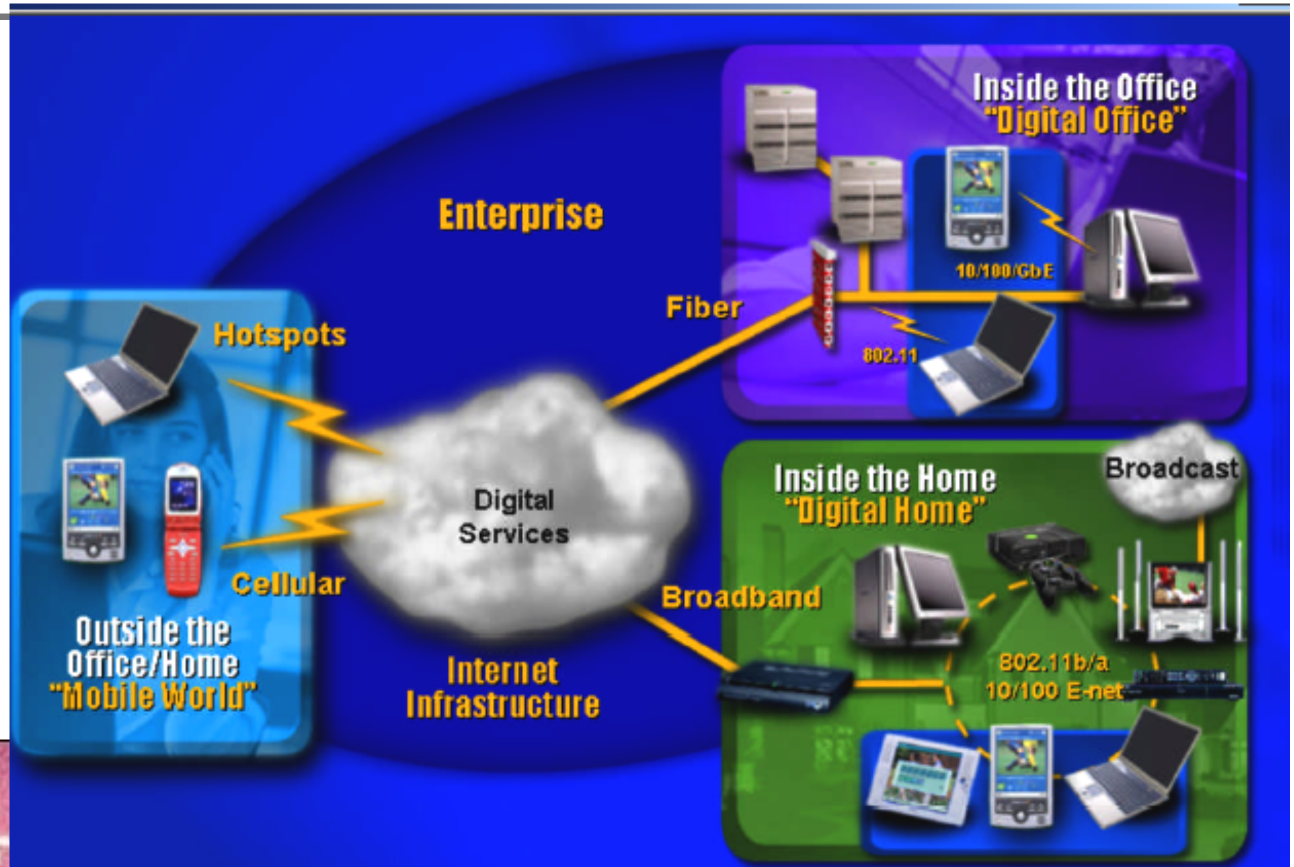


Fuel cell technology

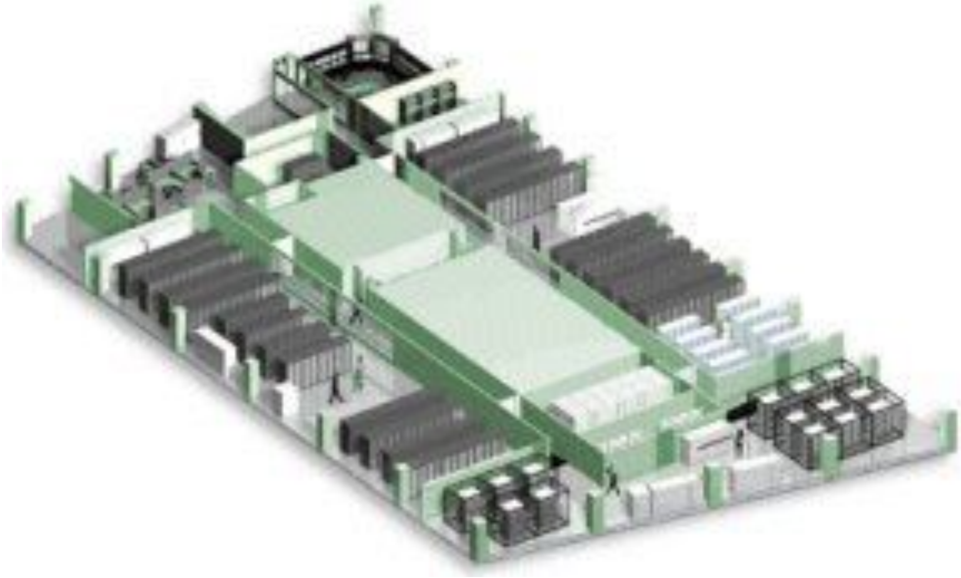




3. Wireless networks are proliferating



4. Data Centers aggregate resources

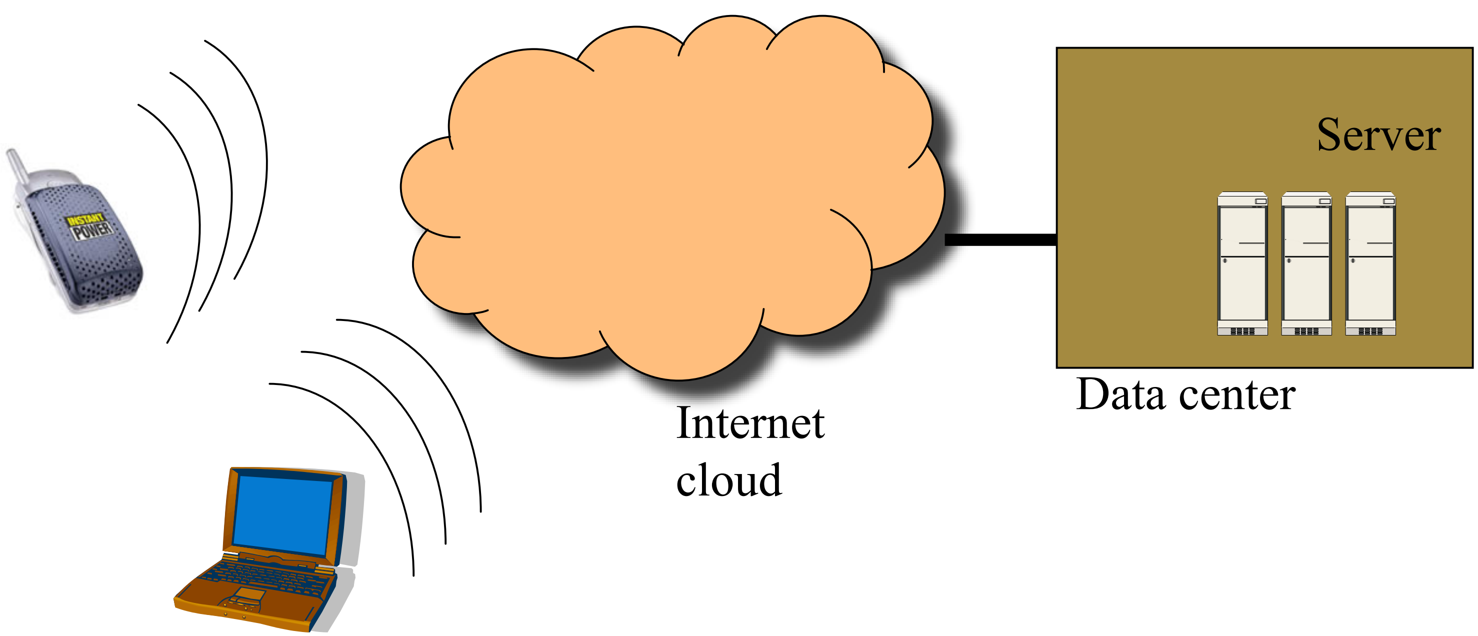


Trends

- Computers getting cheaper and power-aware
- Batteries lasting longer
- Wireless networks proliferating
- Data centers aggregating resources for service hosting

Where will this lead?

- **Ubiquitous mobile devices** will communicate with **resource-rich data centers** over **wireless** and wireline networks



So what?

Use case: thin client

- Task-specific devices at the network edge can leverage ‘heavy’ processing at a data center
- Application examples
  - Voice storage
    - A cell phone can store every word you speak at a data center
    - Can use a multimodal interface to retrieve conversations on demand
  - Image analysis and manipulation
    - A networked camera can shoot digital pictures and upload them to a server
    - Compute-intensive servers can process the image (red-eye reduction, auto-date, translations)

Use case: ‘global’ state

- Provide central view of global state
- Application examples
  - Instant messaging
    - Wireless client can know which ‘buddies’ are online
  - Cargo tracking using RFID
    - Interested end points can get an instant snapshot of location
    - Can run queries on dynamic database (which containers are more than 4 hours behind schedule?)

Use case: coordination

- Central server can coordinate groups of clients
- Examples
  - Form a private network (VLAN) between members of a workgroup
    - Lets users seamlessly participate in a secure collaborative environment
  - Share location information with team members
    - A cell phone or PDA could display the geographical coordinates of team members on a display

Use case: information overlays

- Exploit overlap between realspace and cyberspace to overlay information on physical objects
- Two approaches
  - RFID/Bluetooth
  - GPS
- Application examples
  - Entering an airport updates your PDA to reflect the latest flight information
  - Coming close to a painting in a museum brings up information about it
  - HP Cooltown

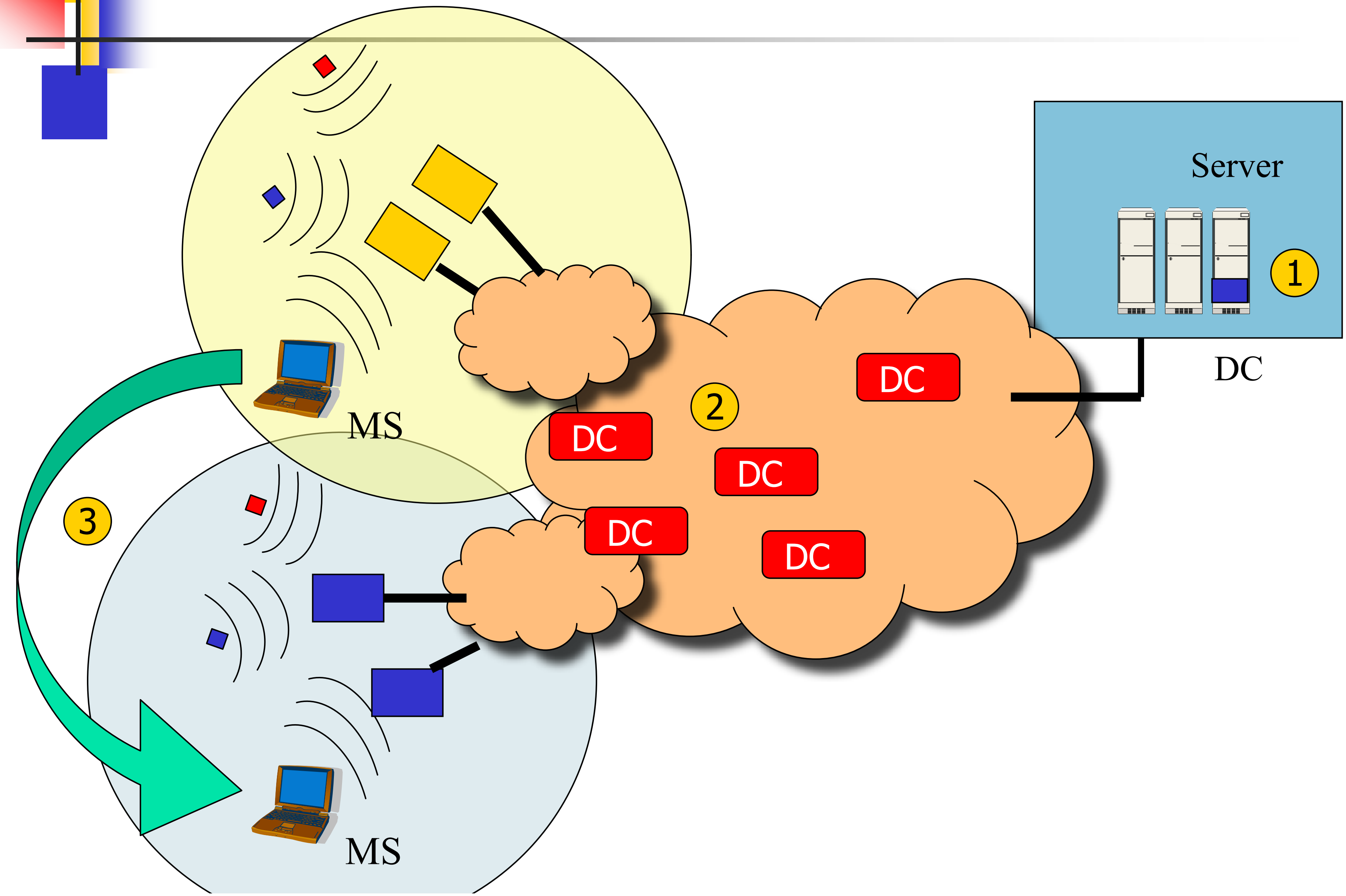
Economic impact

- Applications based on these use cases drive out inefficiencies in production and enhance economic value add
- ROI = Return on Intelligence!

Research areas

- Infrastructure implications of large-scale tetherless computing

- Server virtualization
- Internet Data Center topology
- A hierarchical cryptosystem for fast, secure, roaming between 802.11 hotspots



Server
1
DC
DC
2
DC
DC
DC
DC
MS
3
MS

Server Virtualization

Joint work with P. Goyal, R. Sharma, S. Gylfason, P. Menage, X.W. Huang, C. Jaeger, and T. Bonkenberg

Ensim Corporation

Background

- End systems, including mobile devices, access *services* in data centers
  - A service instance corresponds to an instance of a running application
- Examples
  - Image analysis and transcoding services
  - Coordination and collaboration services
  - Database services
  - Websites

Dedicated servers

- Users or organizations prefer to dedicate a server to a service instance for three reasons
  - Security
    - The service may store sensitive information that should not be seen by others sharing the same server e.g. electronic commerce storefronts
  - Performance
    - The service may require guaranteed CPU, network, memory and disk I/O resources e.g. transcoding
  - Customization
    - The service may need to be customized in a way that precludes its use by other users or organizations e.g. website hosting

Problem

- This doesn't scale!
  - Too many servers
  - Too hard to manage tens of thousands of servers
- Need solutions to
  - Reduce number of servers
  - Make server and service deployment manageable

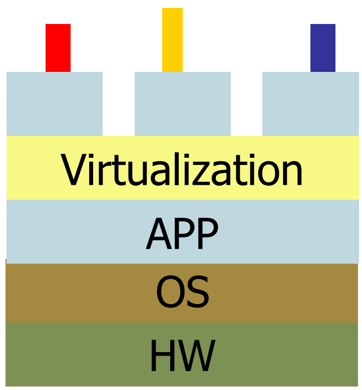
Reducing server count

- Key insight #1: most servers are lightly used
  - If we can pack many service instances on a single server, then can reduce number of servers
- Key insight #2: cannot require application modifications
  - Otherwise no one will use the solution!

Aha – virtualization!

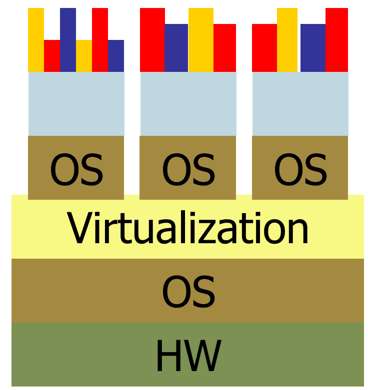
- Virtualization is a standard technique to break the mapping between a service and its implementation
  - Virtualization = interception + indirection + multiplexing
  - Example: virtual memory
- If done properly
  - Doesn’t require any application modification
  - Can provide isolation
  - Can provide performance guarantees
  - Can allow each application instance to be arbitrarily customized

Virtualization approaches



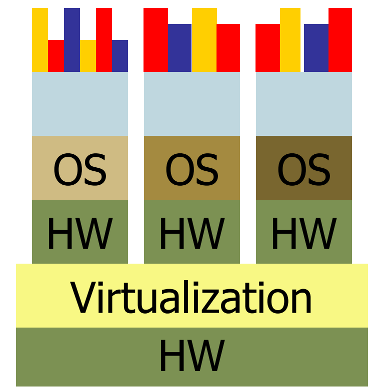
Application virtualization
(Oracle, Apache)

- Requires either re-linking or source-code modification
- Does not provide performance guarantees
- Limits app customization



Private Server
(Ensim)

- No source code or object code changes
- Support for a single OS
- Can provide performance guarantees
- Small overhead



Virtual machine
(VmWare, IBM)

- No source or object code changes
- Allows a single server to host multiple operating systems
- Large overhead
- No performance guarantees

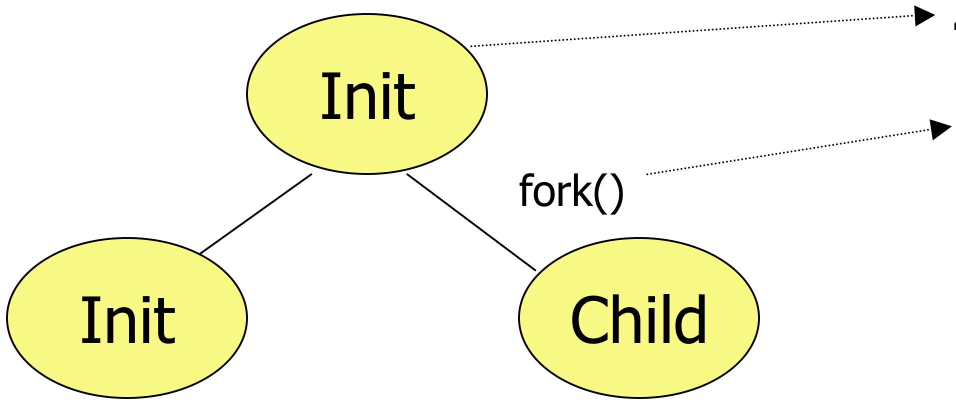
Private servers

- Ensim approach:
  - Virtualize OS interfaces to create *Private Servers (PS)*
  - Each PS appears to be a separate OS instance
  - Each PS is completely isolated from others
  - Does not require modifications to kernel source code
  - PS can run unmodified binaries
- Quality of Service
  - Each PS is guaranteed a resource share in terms of CPU, disk, disk bandwidth, memory, and network bandwidth

Solution overview

- A private server is just a set of processes
- When a process in a PS accesses a shared resource, the access is transparently *intercepted*
- The access is *indirected* to the actual resource with rewritten arguments or rewritten results
- In addition, kernel scheduling is modified to provide resource guarantees to private servers
- 3 key elements
  - Process tracking
  - Access interception and indirection
  - Resource scheduling

Process tracking



PID	PSID
211	1
232	1

Interception and indirection

- Transparently intercept access to **all** system resources, e.g.
  - System calls
  - /proc
  - File system
  - Users, groups, and resources for users and groups
  - Network stack
  - Physical memory and swap
- Two options
  - Filter results of an information query based on PS ID
  - Rewrite the arguments to the call based on *indirection table*

Transparent Interception

- **Essentially based on wrapping system calls**
- To intercept a system call
  - Change the entry function in the system call vector table
- To intercept device access
  - Intercept the ‘open’ system call and parse arguments
- To intercept network access
  - Figure out which file descriptors are for network access, by tracking socket() calls
- To intercept signals
  - Intercept the system calls used to send/receive signals

Indirection

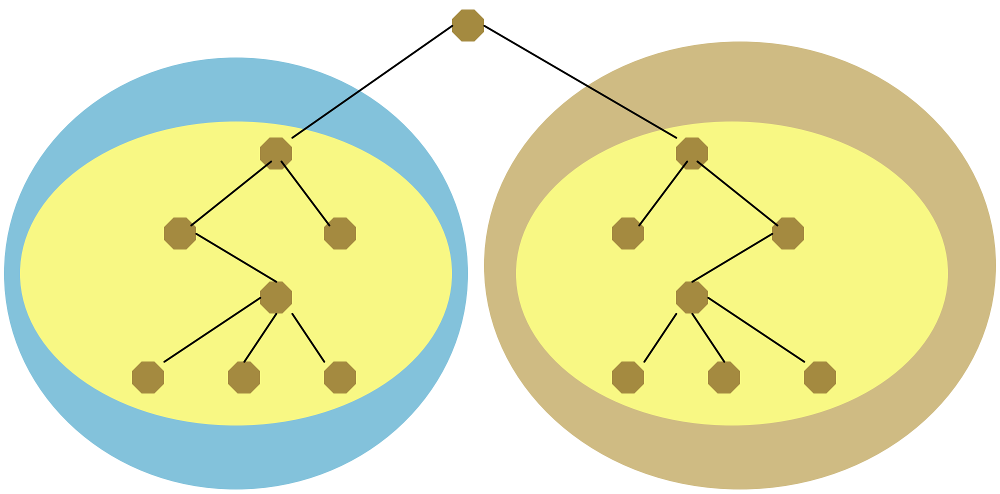
- **Complex: need to do different things depending on what is being intercepted**
- Falls into a few categories
  - Limit actions of the root user(s)
  - Manage process interactions
  - Create an additional level of quotas (user + group + PS)
  - Massage system information
  - Separate network protocol stacks

Limit actions of root users

- Each private server has its own 'root' user
- System calls made by this root user are given greater privileges than system calls by non-root users
- However, even this root user has limits
  - No module insertion
  - Can't browse file system outside of the PS
  - Have permissions only to a specific set of system calls
- Need to parse arguments on every system call and use a table to decide whether it should be allowed or not

Manage process interactions

- Control processes to prevent process interactions (kill, send signal, trace, set scheduling parameters, etc.) from crossing PS boundaries
  - The 'real' root can act on any process
  - Virtual root can act on processes in its PS
  - A parent can act on its children
  - Processes in a PS cannot act on processes in other PSs



Create an additional level of quotas

- Normal Unix has user and group quotas
- Need to add PS quotas
- Done by creating a new file system type whose inodes have the same uid/gid settings as the real file system, but whose quota control operations understand PS quotas
- Quotactl/status calls are intercepted and arguments rewritten to use the new file system
- This allows us to integrate PS quotas seamlessly into the OS

Massage system information

- Create separate syslogs
- Rewrite results of access to /proc
- Limit device access

A separate protocol stack per PS

- Protocol stack code is isolated into a single module and virtualized
- Each PS is given its own module
- Allows very tight control over the network
  - Prevent users from spoofing IP address
  - Fine-grained rate control on packets reads and writes
  - Fine-grained statistics at the application and protocol level
  - Can have a separate firewall for each PS!

Resource scheduling

- Modify schedulers to provide QoS guarantees based on PS ID
  - Hierarchical Start-time fair queueing for rate allocation
  - Leaky bucket for rate control

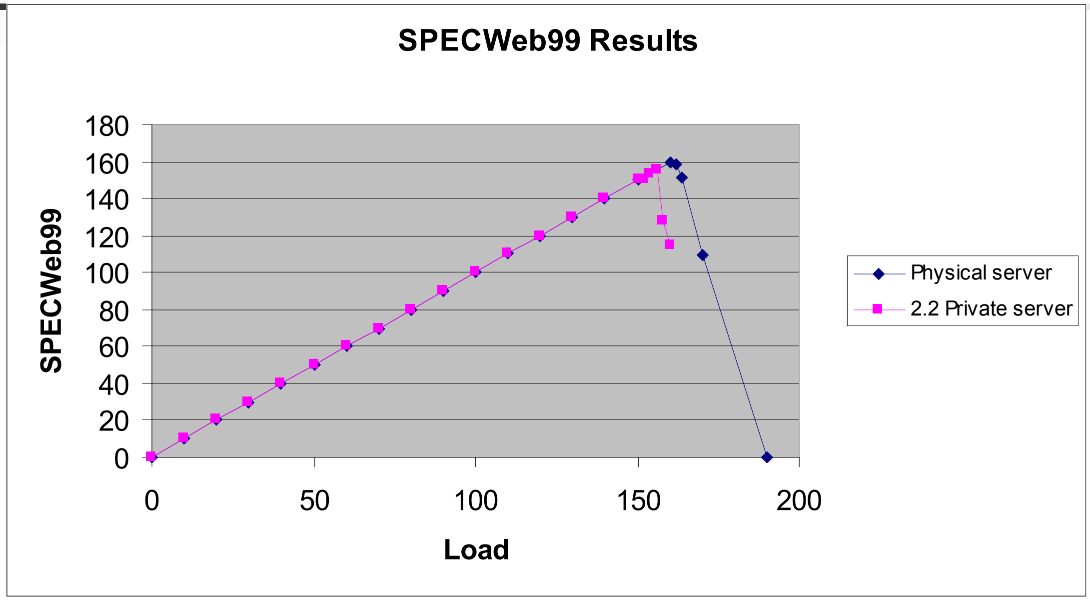
Net result

- A process in a private server
  - Has its own file system
  - Can run any application with unmodified binaries
  - Has guarantees on CPU, network, memory, disk quota, disk I/O rate
  - Cannot see external processes
  - Cannot send signals to other PSs
  - Has a unique ‘init’ parent
  - Has limited access to devices
  - Has a unique IP address and cannot spoof IP addresss
  - Has unique users and groups
  - Supports a ‘virtual’ root
  - Limits ioctls
  - Can only snoop local packets
  - Has access to most of /proc
  - Can configure its own protocol stack

Performance

- Additional kernel memory per private server: 200K
  - Very small footprint
- Additional disk space per private server: 300MB
  - To recreate the base file system
- Number of private servers/physical server: up to 90
- Private servers in production use: about 4000

Performance - continued



Response time with physical servers at peak load (ms)	Response time with a 2.2 private server at peak load (ms)	Overhead
336	343	2.04%

Domain type	Operations/sec	Response time (ms)	Bitrate (bps)
Low 1	6	2887	42088
Low 2	7	2811	42585
High 1	14	1412	85686
High 2	14	1416	84780

Consequences

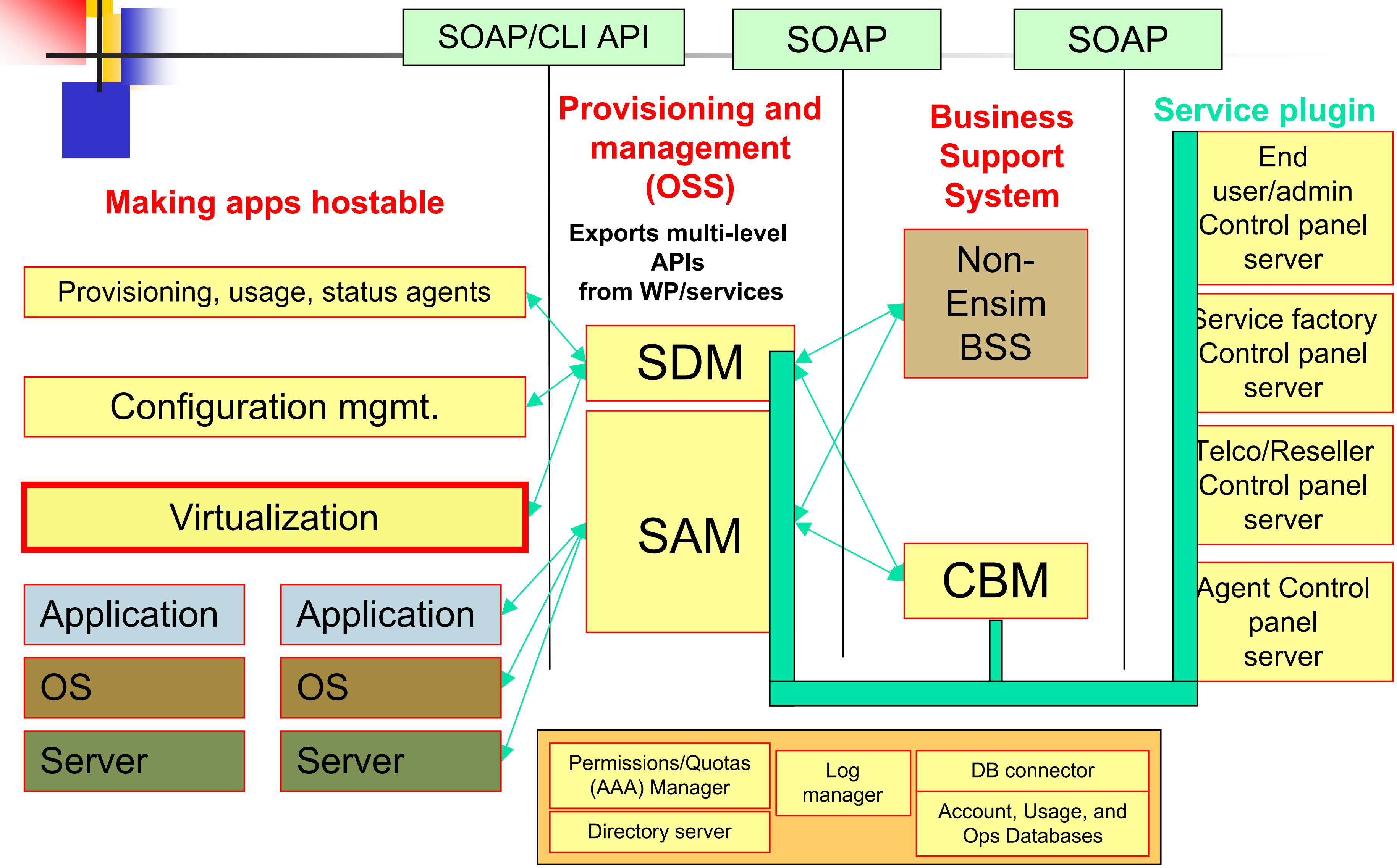
- Allows a datacenter to offer a service on a virtual server to an organization
- Service can be arbitrarily customized
- Services can be given performance guarantees
- Services are run in a secure environment
- Services can be densely packed
- Freebie: resource allocation to a service can be dynamically modified
  - First steps towards ‘grid computing’

Related work

- Vserver
  - Uses security contexts for identifying each PS
  - Security context checking has to be hacked into kernel
  - Hard to do without modifying source
  - No support for QoS (yet)
- Virtual machine architecture (IBM, VMWare)
  - Has a heavy resource/performance overhead
- Isolation microkernel (Denali, Xeno)
  - Does not support commercial OS
  - Requires extensive rewrite of OS internals to match microkernel API
- Resource containers, restricted execution contexts, virtual services
  - Share components between virtual servers
  - Complex programming abstraction, complex policies
  - Very hard to manage

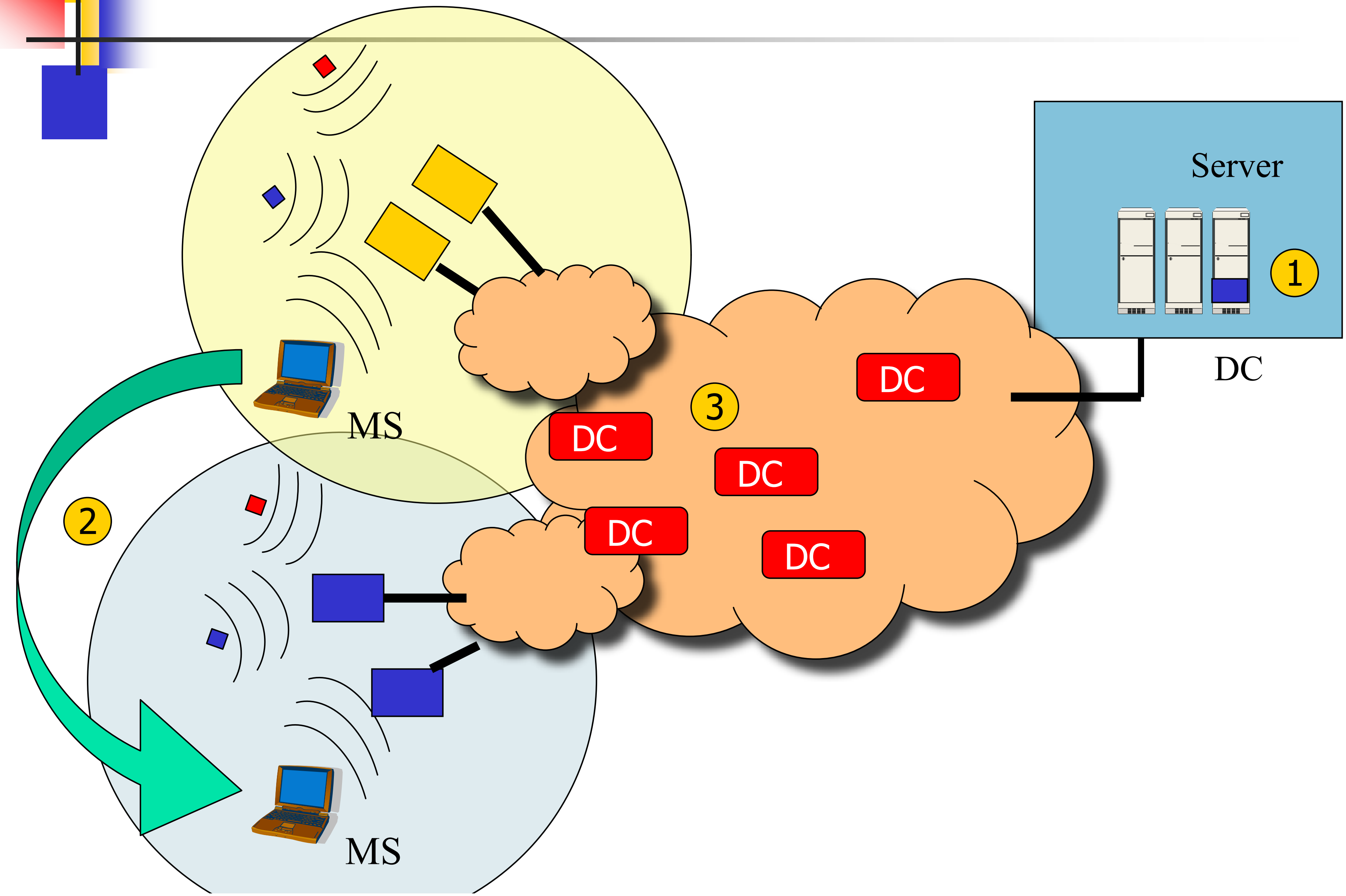
But this is only 5% of the story…

Server and service management



A hierarchical cryptosystem for fast secure roaming

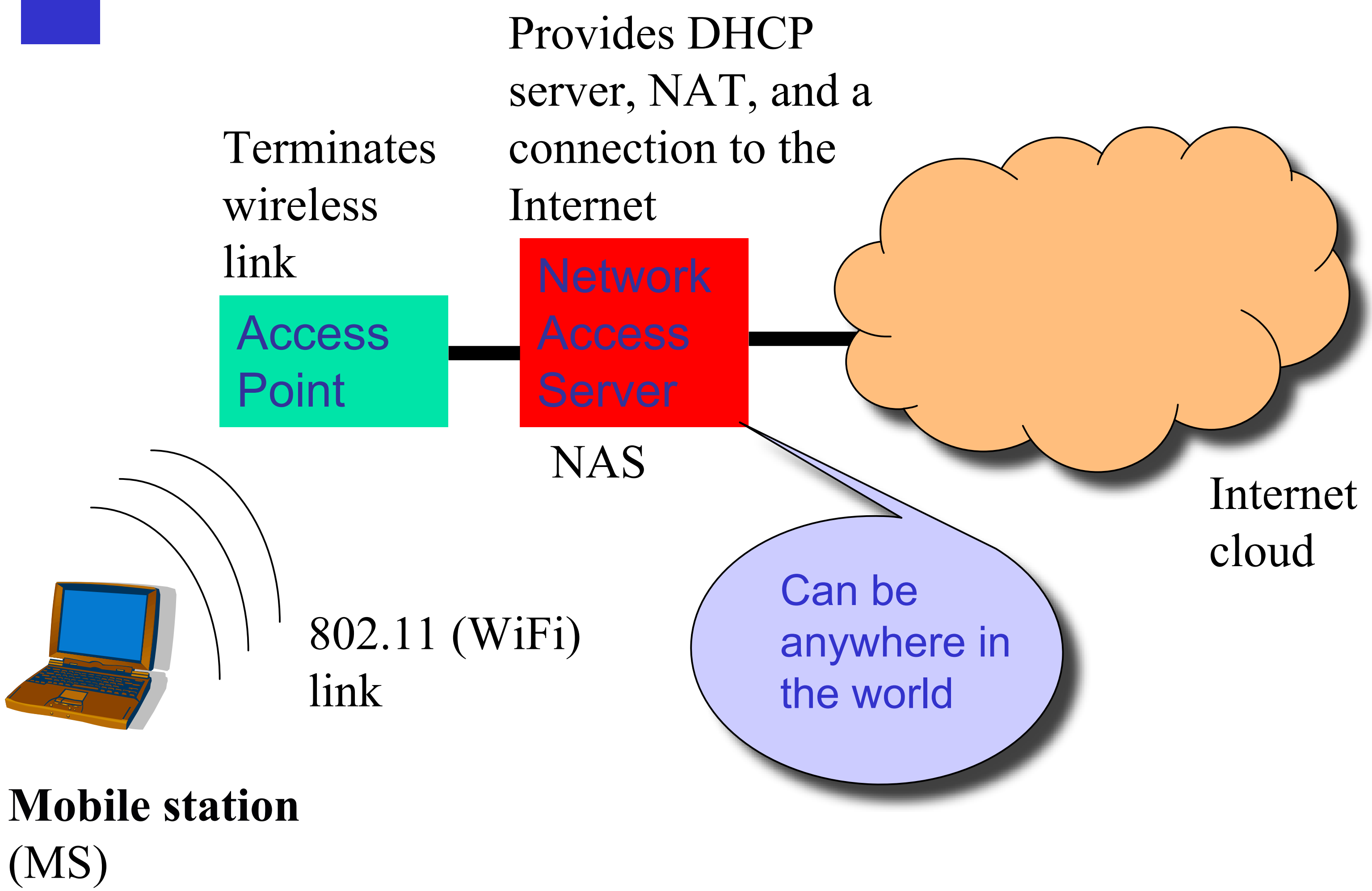
Joint work with C. Nagarkar and M. Kopikare
Stanford University


Server
1
DC
DC
3
DC
DC
DC
DC
MS
2
MS

Outline

- WiFi basics
- Security and authorization in WiFi networks
- Intra-federation authorization and handoffs
- Issues in inter-federation authorization
- WASSUP architecture
- WASSUP features
- Summary

WiFi basics


Provides DHCP server, NAT, and a connection to the Internet
Terminates wireless link
Access Point
Network Access Server
NAS
Internet cloud
Can be anywhere in the world
802.11 (WiFi) link
Mobile station
(MS)

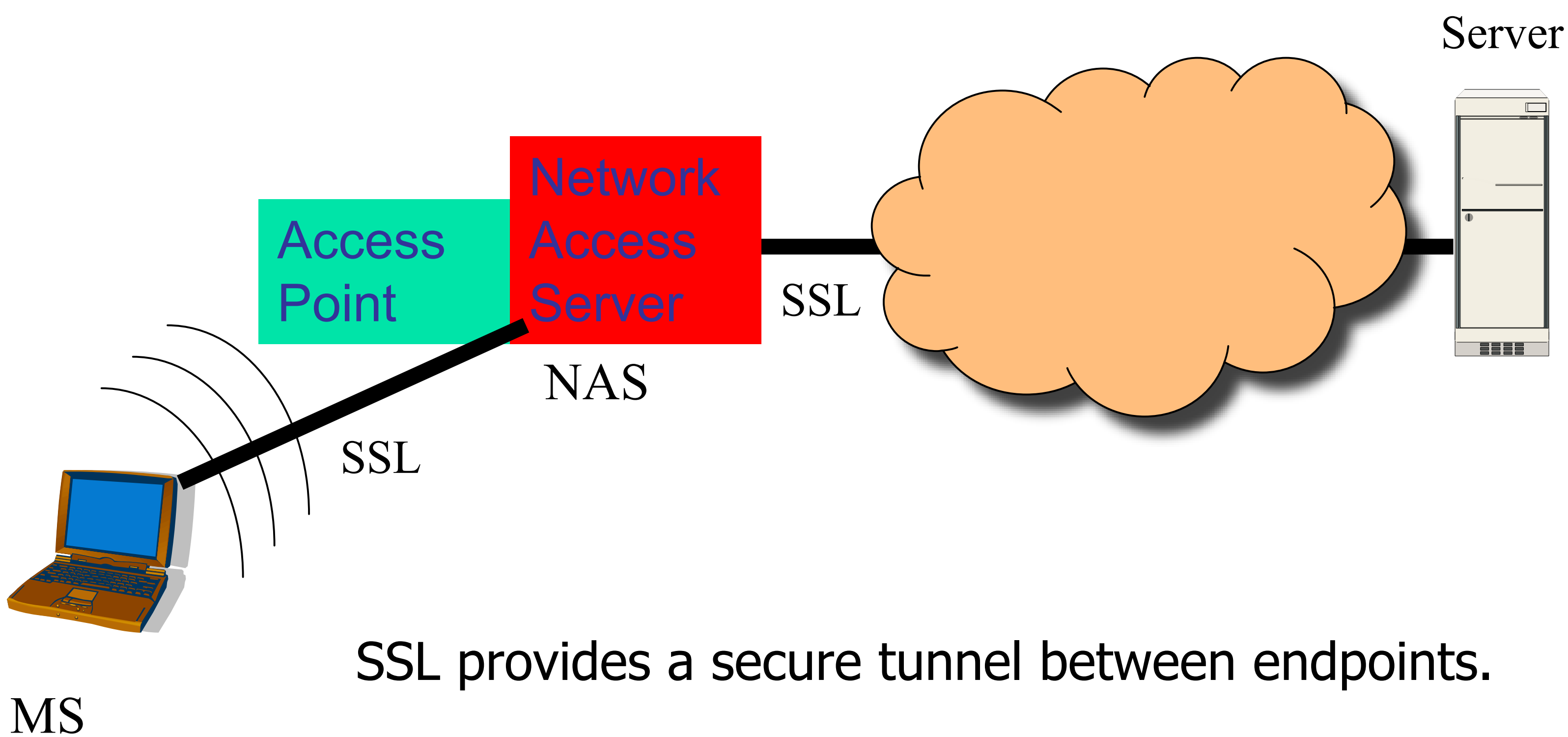
802.11 networks abound

- Approximately 10,000 hotspots worldwide today
  - 1605 hotspots listed at http://www.wifinder.com
  - Boingo has 800 hotspots
  - T-Mobile has1696 hotspots
- Intel, IBM, Verizon have announced Project Rainbow with plans for 1000s of hotspots
- IDC projects 40 million WiFi users in 2006

Issues: security and authorization

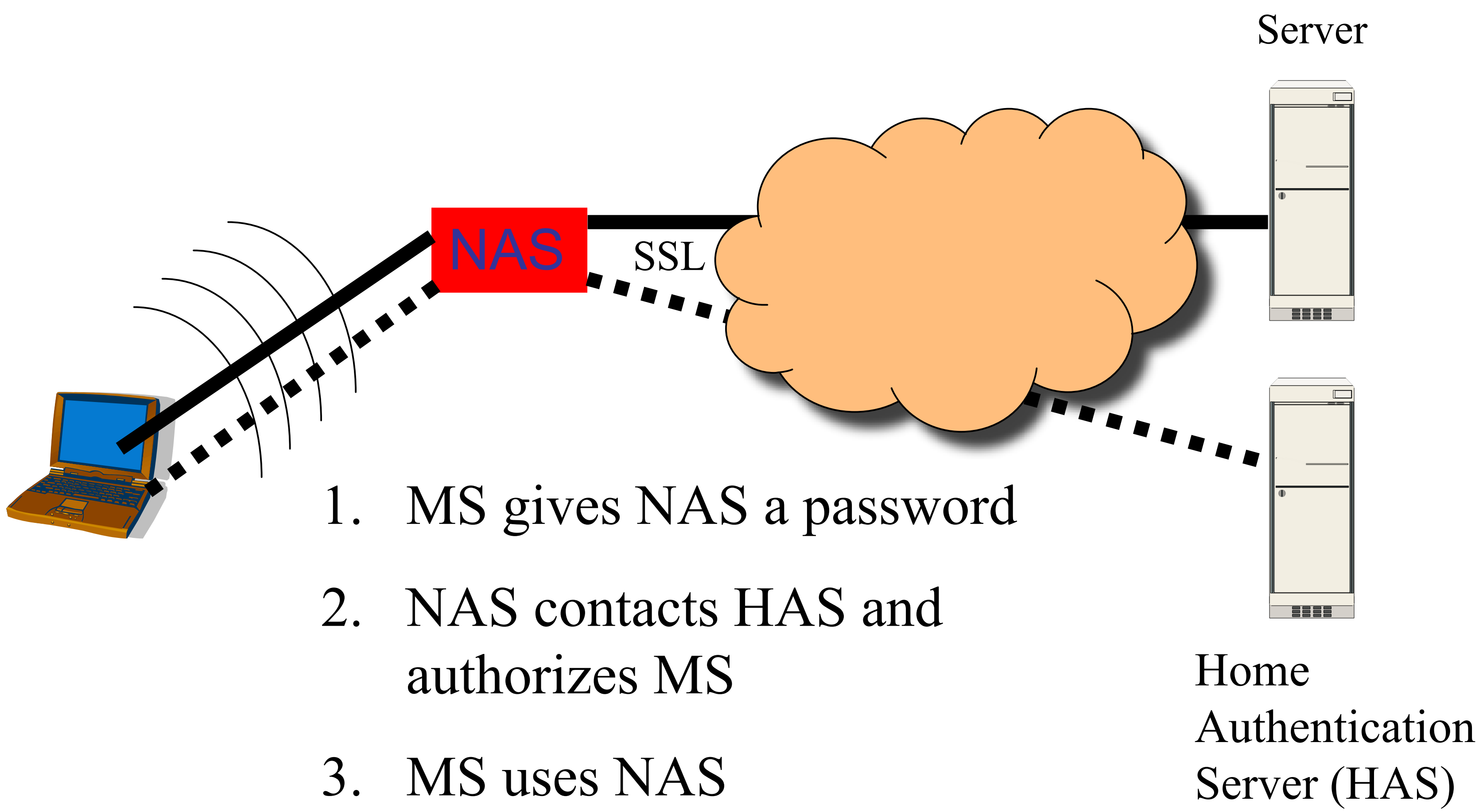
- Can a mobile station be sure that its data is private?
  - If you log in to Wells Fargo from a coffee shop in Costa Rica, should you worry?
- Can the 802.11 network be sure that only valid mobile stations are using it?
  - Corporate intranets dislike unauthorized use

A naïve security solution



SSL provides a secure tunnel between endpoints.

Naïve authorization



1. MS gives NAS a password
2. NAS contacts HAS and authorizes MS
3. MS uses NAS

Life is not so simple!

- What if the NAS is a rogue?
  - Can intercept all non-encrypted traffic
  - Worse, it can pretend to be a server, terminate SSL, and then intercept passwords (man-in-the-middle attack)
    - Any website can be spoofed!
  - Can allow unauthorized mobile stations to access the network

MS, NAS, and HAS must mutually authenticate each other

Mutual authentication

- Can be done in many ways
- Current standard is IEEE 802.11X which allows for Extensible Authorization Protocol (EAP)
- EAP allows any mutual authentication scheme to be plugged in
- A common scheme is standard Unix-style passwords
- Secure Remote Protocol (SRP) is **much** better

SRP for mutual authentication

- SRP is a clever way to use a simple password for mutual authentication of two entities
- Does not require Public Key Infrastructure
- Can be used to set up a session key
- As long as server keeps password file secret, can guarantee a secure channel and mutual authentication

SRP basics

MS | **HAS**

Stores username, and verifier = hash (password)

Send user name and hash(nonce1) →

Replies with hash (verifier, nonce2) ←

Computes Key from nonce1, password and hash (verifier, nonce2)

Computes Key from nonce2, verifier, and hash (nonce1)

Send challenge = hash(Key, …) →

Verifies and replies with response = hash(Key, …) ←

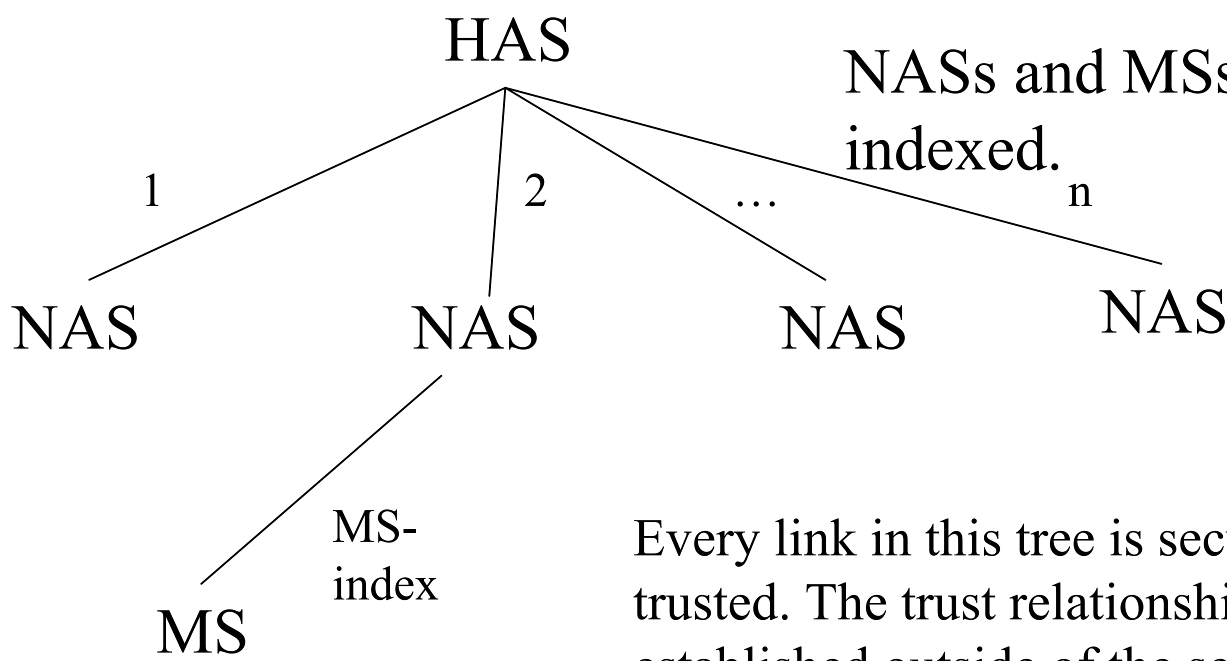
Verifies

On success, MS and HAS are mutually authenticated

How does SRP help?

- Mobile station and HAS can mutually authenticate each other (password is the shared secret)
- Anyone who trusts the HAS can be told about entities that the HAS trusts
- Suppose that NAS establishes a secure channel with HAS when it becomes part of the federation
  - HAS can give NAS a credential that NAS and MS can use to mutually authenticate each other

The Authorization Tree



NASs and MSs are indexed.

Every link in this tree is secure and trusted. The trust relationship is established outside of the security framework. Trust is transitive.

Solving the rogue NAS problem

MS | **NAS** | **HAS**

Stores username, and verifier = hash (password)

Authenticate HAS using SRM. Establish K(MS-HAS)

Authenticate MS using SRM. Establish K(MS-HAS)

Ask NAS for NAS-index and credential

Tell NAS **credential = Hash(K(MS-HAS) + C*NAS-index)**

Compute K(MS-NAS) = Hash(credential + C*MS-index). Tell MS NAS-index

Compute K(MS-NAS) = Hash(Hash(K(MS-HAS) + C*NAS-index) + C*MS-index). Challenge NAS with Hash (K(MS-NAS)…)

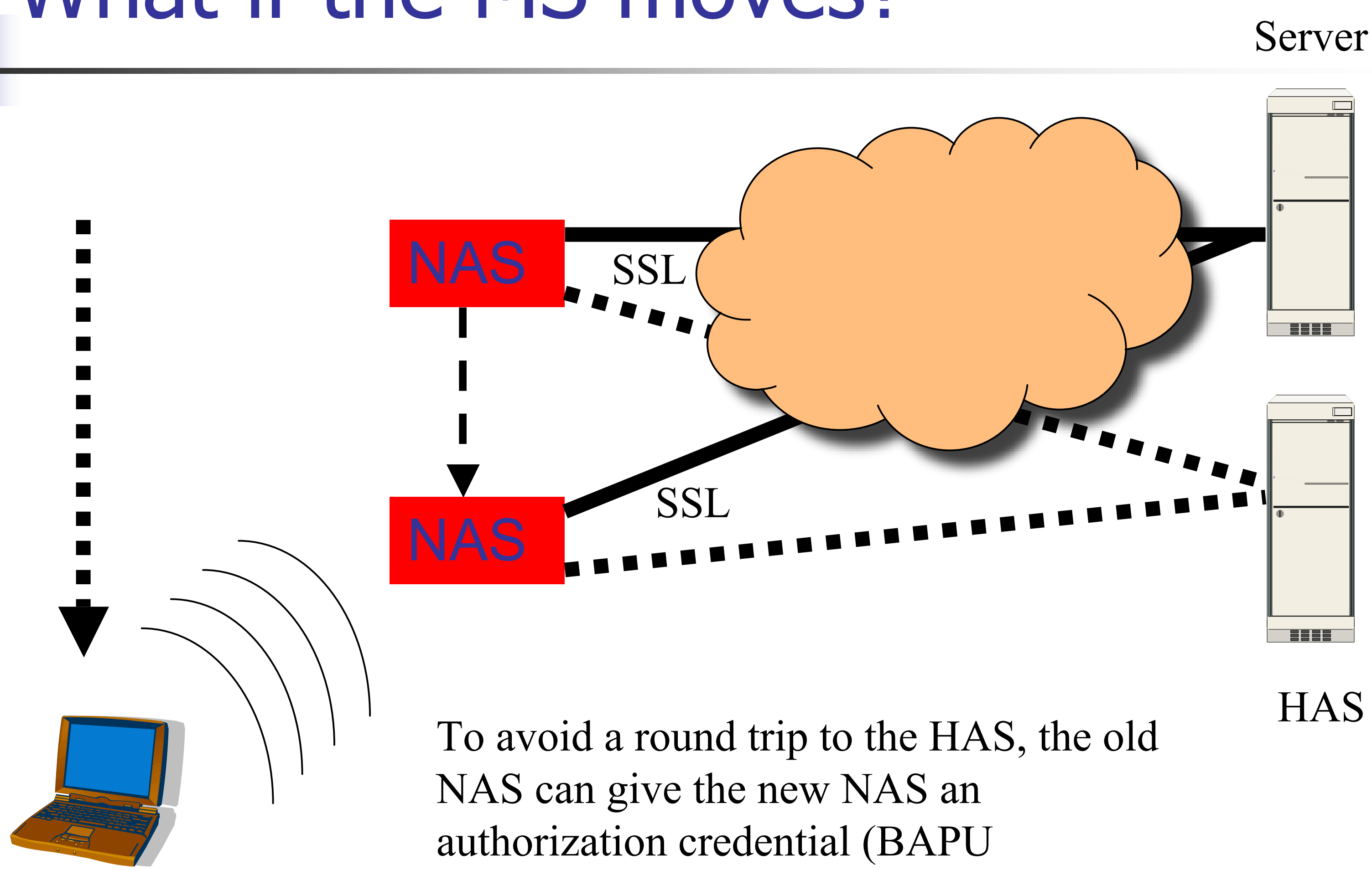
Compute and verify Hash(K(MS-NAS), …) and respond

Verify

Solving the rogue NAS problem

- This is basically an extension to SRP
- It can be shown that this scheme is cryptographically secure
- For properly chosen values of Hash and C, MS can verify that credential came from a valid NAS with very high probability

What if the MS moves?

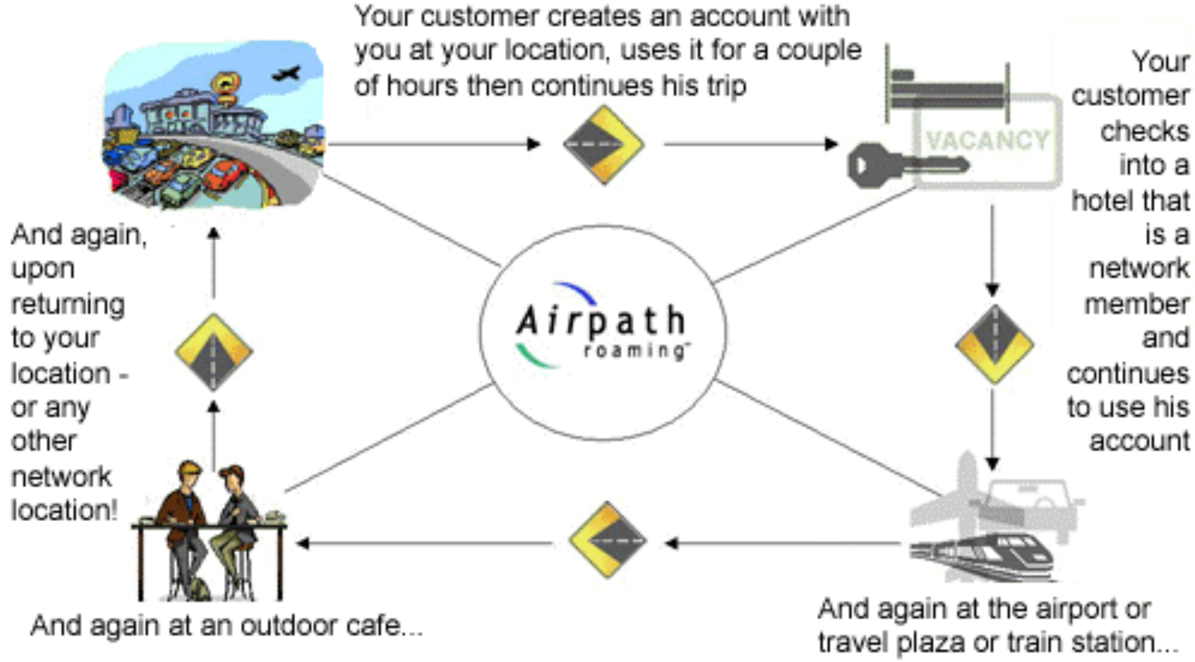


To avoid a round trip to the HAS, the old NAS can give the new NAS an authorization credential (BAPU scheme). The MS may also need to acquire a new IP address or use Mobile IP to tell its Home Address Agent about its new location.

Federations

- The description so far allows sets of NASs and a HAS to work together to mutually authenticate MSs.
- This forms the basis for a federation
- Handoffs within a federation are fairly straightforward
  - BAPU scheme optimizes handoff
- This has made federations commercially feasible

A commercial example



Graphic © Airpath

Federations abound

- An incomplete list
  - Boingo
  - Airpath
  - T-Mobile
  - Pass-One
  - Megabeam
  - Telia Mobile
  - iPass
  - Sputnik
- Most future access points will have to belong to one or more federations to amortize the cost of marketing and customer acquisition

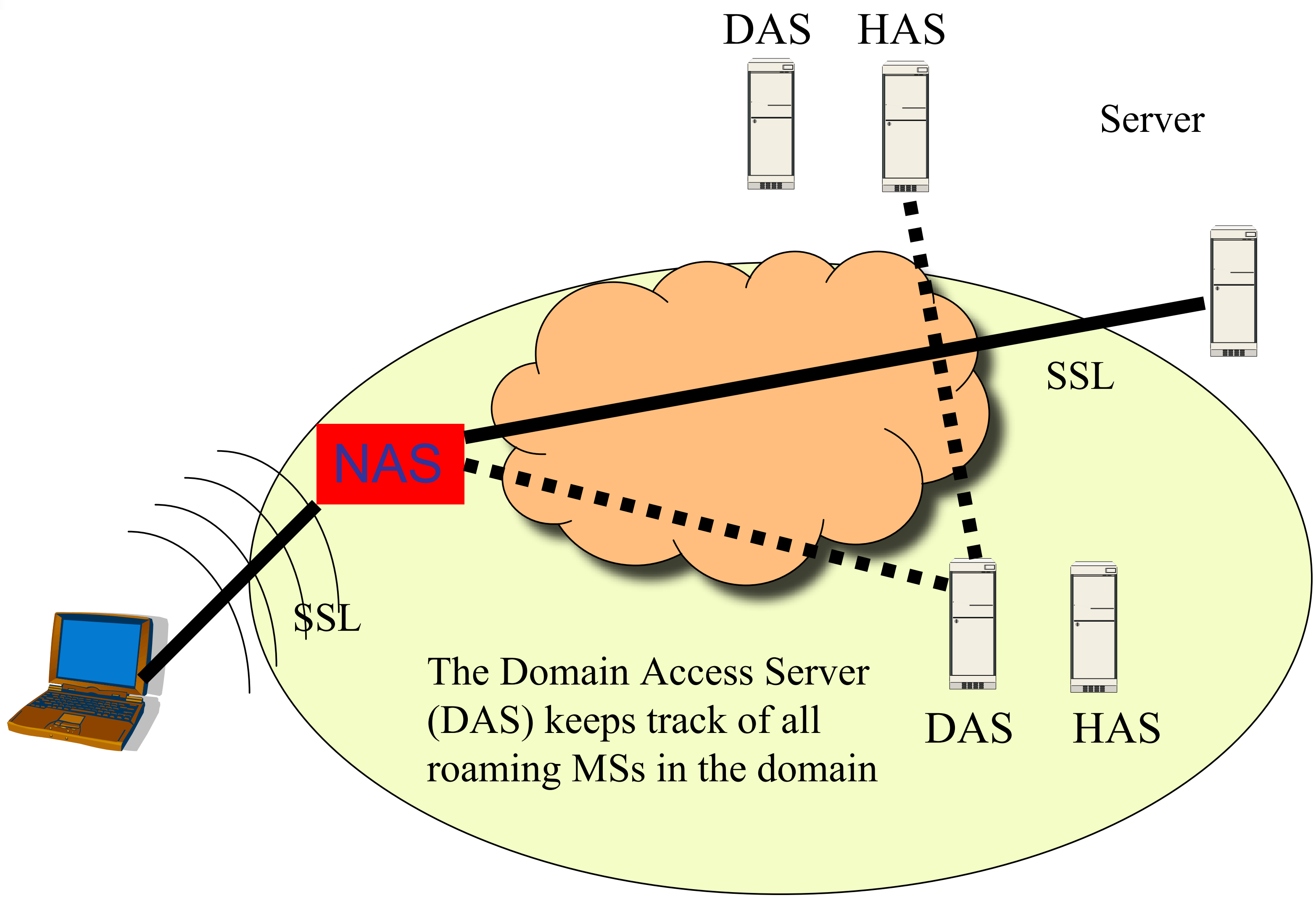
But…

- It is very unlikely that all WiFi subscribers will want to belong to the same federation
- What if a subscriber belongs to one of the federations and wants to roam to another?
  - How to authorize a roaming MS?
  - How efficient is an inter-federation handoff?
  - How can the roaming service provider get paid?
  - If a NAS is compromised, how much damage can it do to the system?

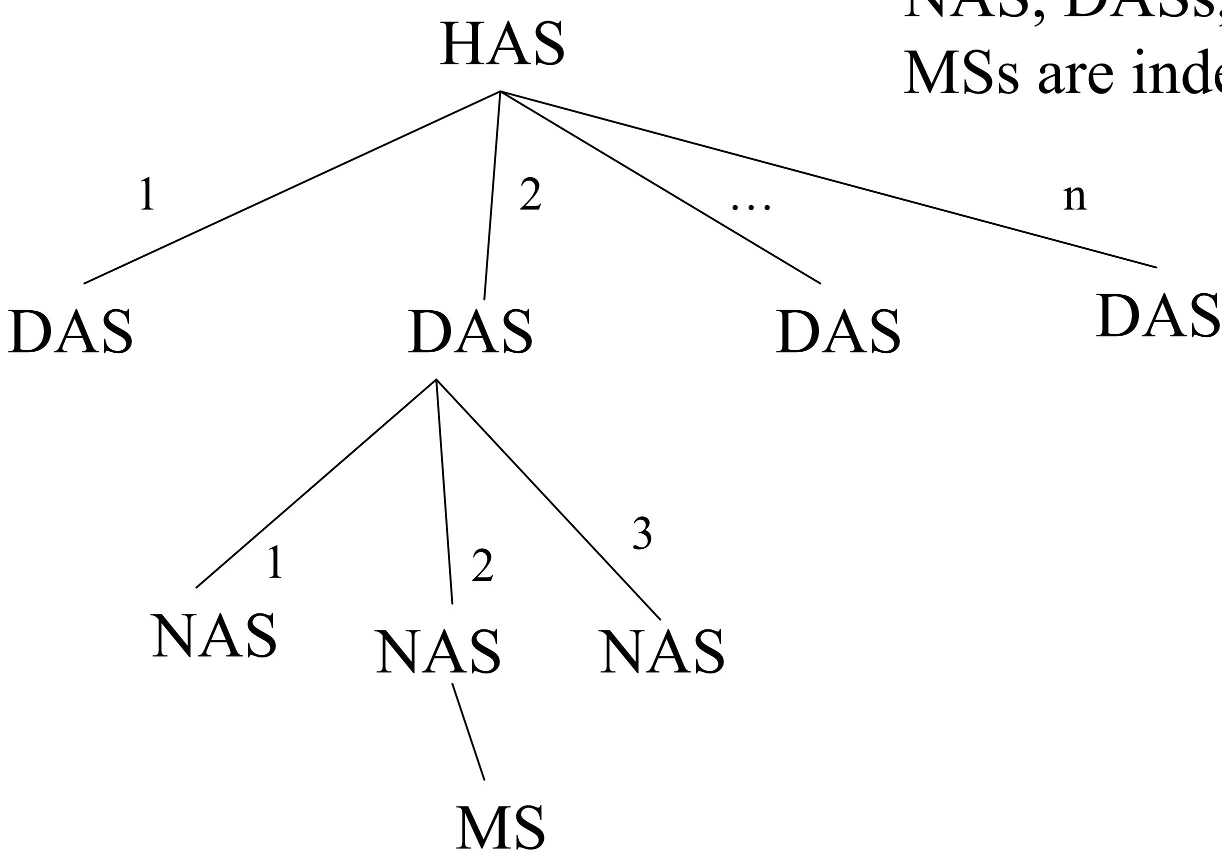
WASSUP

- Wireless Access with Secure, Scaleable and Ubiquitous Performance
- Provides solutions for inter-federation roaming and fast, secure, inter-federation handoffs
- Also provides authorization for roaming users and non-repudiable billing
- Robust: limits damage from a compromised NAS

WASSUP Architecture



New Authorization Tree



NAS, DASs, and MSs are indexed.

Authorizing a roaming MS

- Each HAS establishes a trust relationship with all other DASs
- Each DAS establishes a trust relationship with every NAS in its domain
- MS mutually authenticates its own HAS using SRP
- Now, repeat credential exchange twice
  - HAS gives DAS a credential
  - DAS gives NAS a credential
- By knowing the index of DAS and NAS, MS can compute the credential and challenge the NAS
- NAS verifies and responds with a key computed with its credential
- This mutually authenticates MS, NAS, DAS, and HAS

Solution in more detail

MS **NAS** **DAS** **HAS**

HAS: *Stores 'salt', username, and verifier = hash (password)*

MS: Authenticate HAS using SRM. Establish K(MS-HAS)

HAS: Authenticate MS using SRM. Establish K(MS-HAS)

HAS: Tell DAS Hash(K(MS-HAS) + C*DAS-index)

MS: Ask NAS for indices

DAS: Tell NAS credential = Hash(Hash(K(MS-HAS) + C*DAS-index) + C*NAS-index)

NAS: Compute K(MS-NAS) = Hash(credential + C*MS-index). Tell MS DAS-index, NAS-index, and MS-index

MS: Compute K(MS-NAS) = Hash(Hash(Hash(K(MS-HAS) + C* DAS-index)+ C*NAS-index) + C*MS-index). Challenge NAS with Hash (K(MS-NAS)…)

NAS: Compute Hash(K(MS-NAS), …) and respond

MS: Verify

Inter-federation roaming

- This solution can be further generalized
- Can construct a hierarchy of servers between HAS and MS
- Once MS and HAS are mutually authenticated, credentials can be **chained** to authenticate every element in the path
- Key-chaining is a novel contribution of WASSUP that is a general technique applicable to other cryptosystems

How about fast handoffs

- When a mobile moves from a NAS belonging to one federation to a NAS belonging to another federation, there can be substantial delays
  - Have to validate entire NAS-DAS-HAS path
- Can we optimize this?

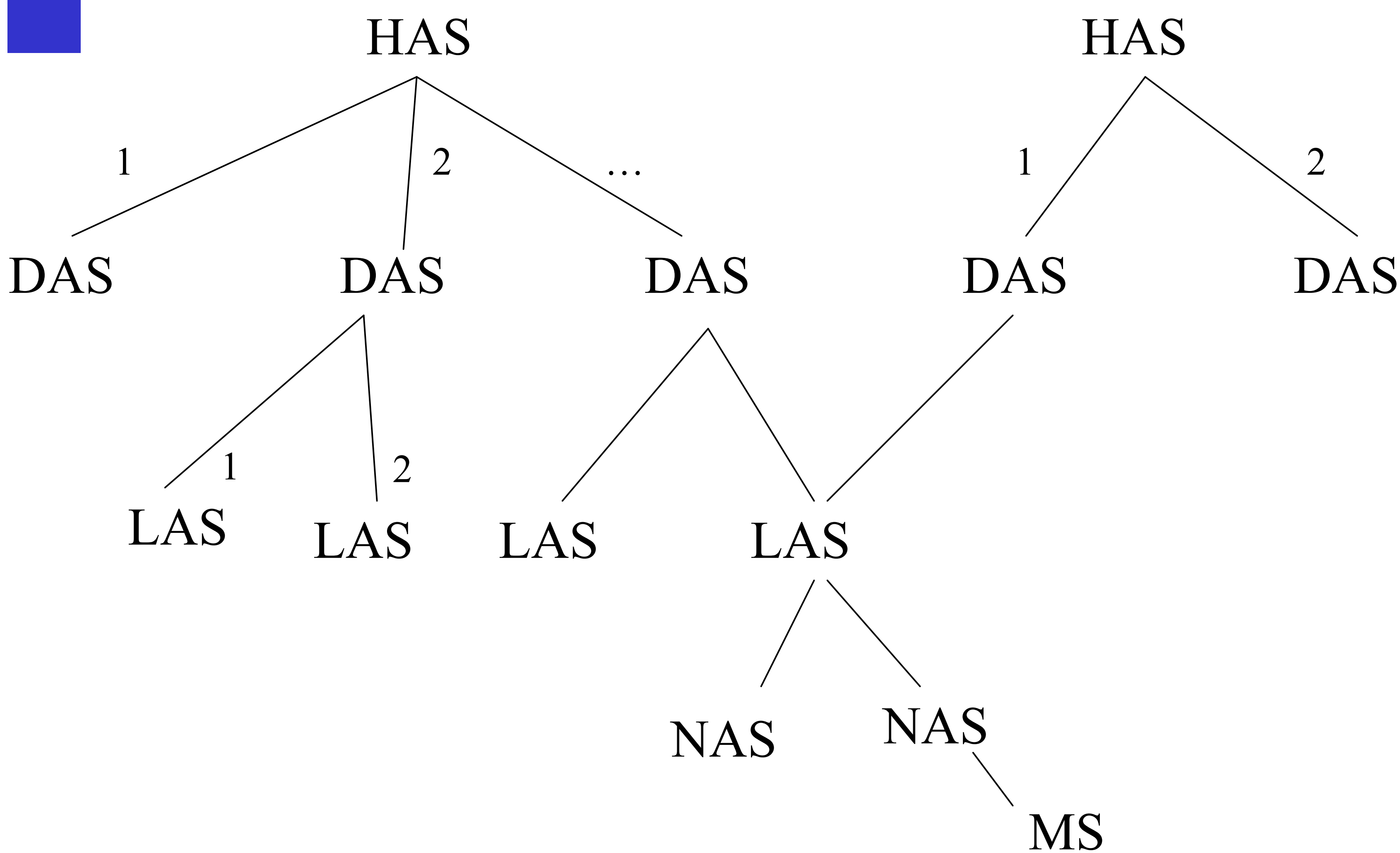
Consider a use case

- Talking on your WiFi mobile as you walk through a mall
- Every store could belong to a different federation
- You will be handed off from one NAS to another
- But may incur substantial delays each time
- Can we exploit locality?

Local Authorization Server

- A LAS is an authorization server that is shared among multiple federations
- It is trusted by multiple DASs
- It sits in the authorization tree between a DAS and a NAS
- NASs from multiple federations can get a chained credential from the local LAS
- So, if a MS moves between NASes within the same federation, or moves back and forth between the same set of federations at a single location, there is no need to contact the HAS
  - Reduces latency

WASSUP Authorization Tree



LAS benefits

- LASs reduce handoff latency for handoffs between NASs belonging to the same set of federations
  - But it doesn’t reduce the first time authorization latency
- Leverages the key chaining algorithm
- Can also provide a single DHCP server for a set of NASs, to reduce overheads from Mobile IP

WASSUP features

- Fast and secure inter-federation roaming
- Rapid, simple, rekeying
- Integrated with usage accounting system
- Robust against attacks
- Easy to integrate with existing infrastructure

Rekeying

- If an MS wants to rekey, it simply asks the NAS to change its MS-index
- This changes K(MS-NAS)
- K(MS-NAS) provides over-the-air encryption for privacy

Accounting

- When an MS is authorized, NAS knows MS UID, and its IP address
- Can trivially account for MS's bandwidth usage
- Reports this to DAS to consolidate billing for roaming access
- What if DAS is untrustworthy?
  - It can bill a MS even with no usage!

Accounting: Solution 1

- When an MS accesses a domain, it is asked to digitally sign an *undertaking* its private key
- Undertaking contains
  - MS UID
  - DAS UID
  - Current time
  - Usage time period
  - Traffic rate
- DAS verifies and stores the undertaking and presents it to the HAS for billing
- This guarantees non-repudiable billing
- However
  - Overhead for verifying the undertaking on every handoff
  - Overhead for storing the undertaking
  - What if the MS moves away before the time period expires
    - Will still get billed!

Accounting: Solution 2

- Solution 1 is overkill
- Rely on social/legal pressures to enforce billing accuracy
- If an MS user is wrongly billed, they will complain
- If a HAS gets a lot of complaints about a particular DAS, they can break the trust relationship
- This is probably more realistic

WASSUP robustness

- Basis of robustness is key chaining
  - All keys are derived from a single K(MS-HAS) master key
- Only secret information is password file at each HAS
  - Even if this is stolen, the only possible attack is man-in-the-middle, which is much harder than identity theft (I.e. if raw passwords are stored at HAS)
- Attacks on DAS, LAS, and NAS cannot compromise authorization and privacy unless the MS is a complicit party
- A hacked NAS can, at most, generate false billing records
- A hacked NAS will not give the hacker access to any other NAS, or any other part of the system
- If K(MS-NAS) is broken, simple rekeying will change the key in a way that cannot be 'followed'

Integrating WASSUP

- IEEE 802.11X allows SRP to be plugged in as an EAP
- A federation needs a way to recognize the HAS for a non-local UID
  - Federations allowing roaming access need to specify a global UID space (can just be UID@federation)
  - Existing HAS can then serve as a WASSUP DAS
- LASs can be added incrementally to improve performance

Related work

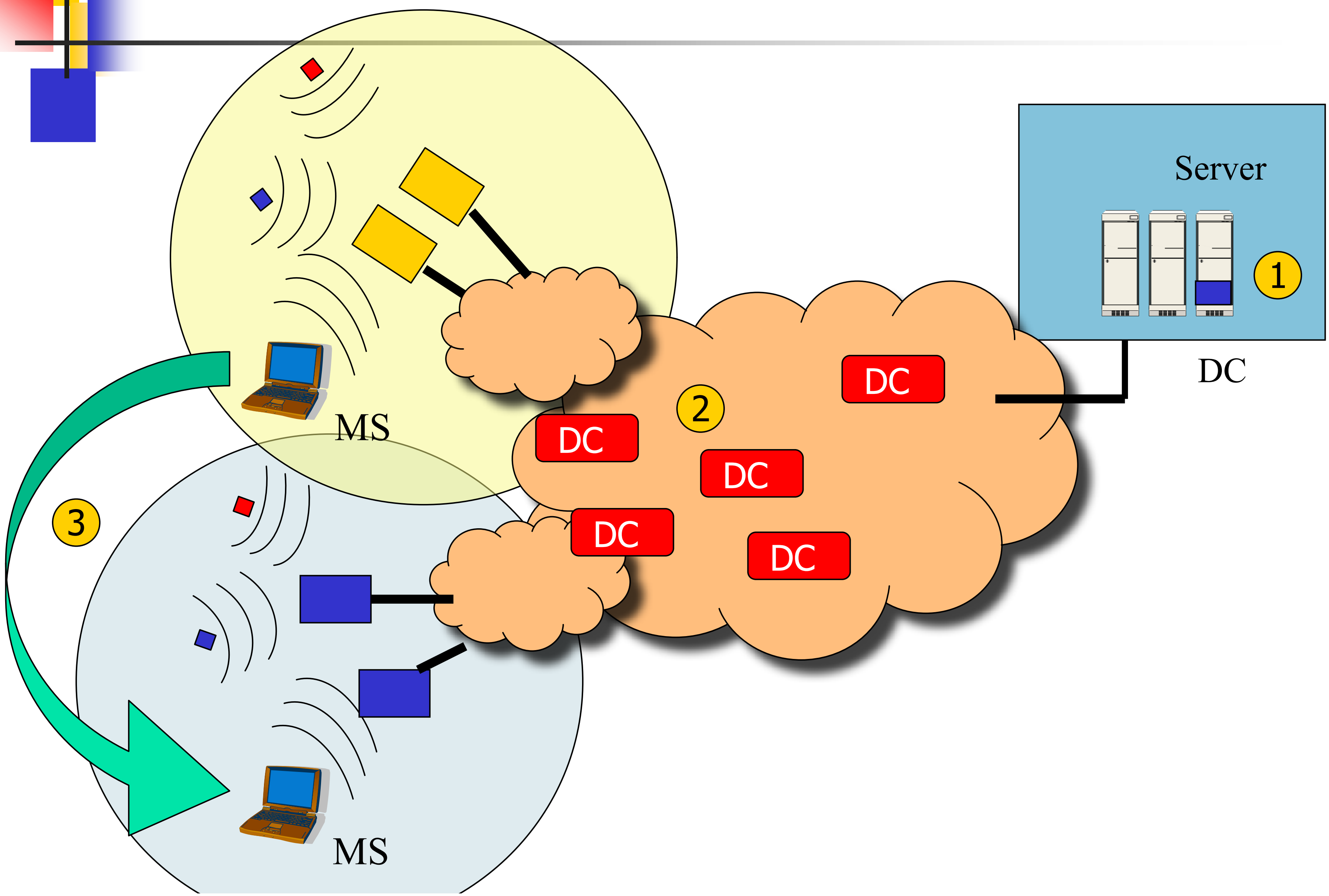
- Hierarchical cryptosystems
  - Do not explicitly support caching and multiple federations
- Security for nomadic systems
  - Solve a harder problem (disconnected authorization) not relevant here
- Multicast group security systems
  - Solve a related problem, but focus on keeping excluded members out
- PKI systems
  - Much heavier weight
  - For mutual authentication, require users to obtain key pairs

Summary

- WiFi networks are mushrooming
- Security and authorization are critical (and distinct) issues
- Existing solutions allow formation of federations, but do not address inter-federation roaming, and fast handoffs
- WASSUP provides a simple, robust, and efficient architecture for inter-federation roaming and hand offs
- Can be integrated into existing architecture with little effort

Internet Data Center Discovery

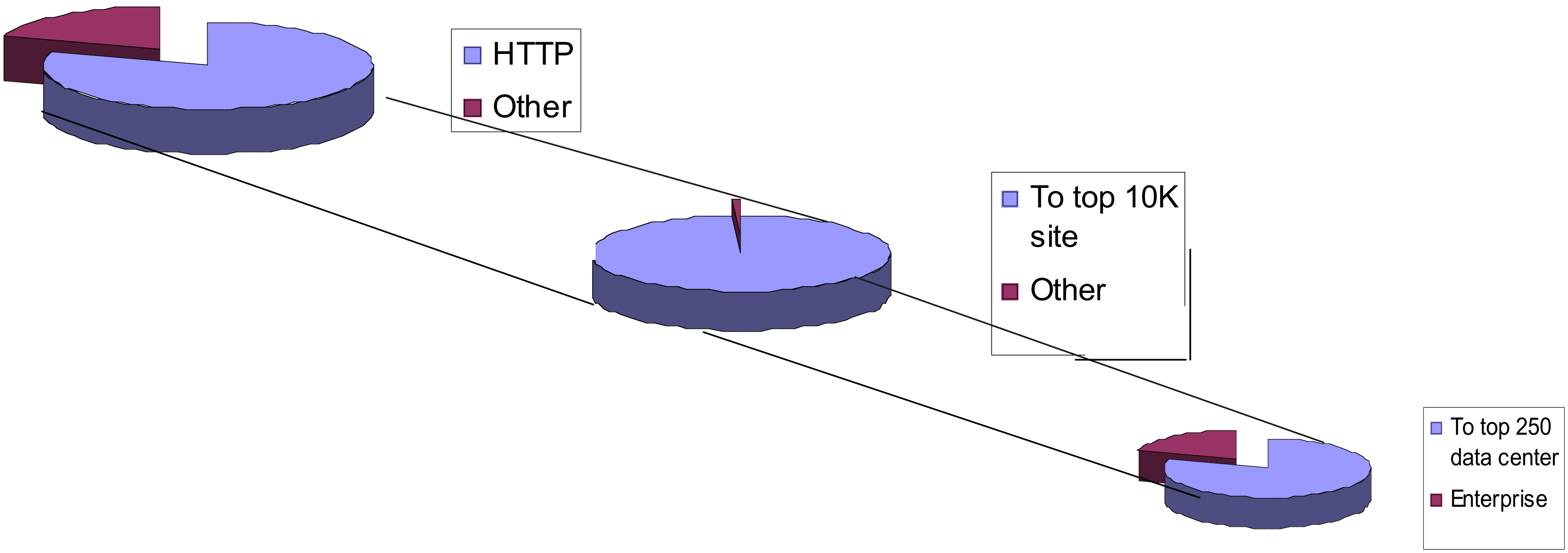
Joint work with R. Govindan (USC), A. Jain, and G. Kwatra (IIT, Delhi)


Server
1
DC
2
DC
DC
DC
DC
DC
MS
3
MS

Internet Data Centers

- Datacenters represent a rich aggregation of computing resources
- Highly connected to the Internet backbone
- Hypothesis
  - Most wide-area Internet traffic is going to data centers

Hypothesis

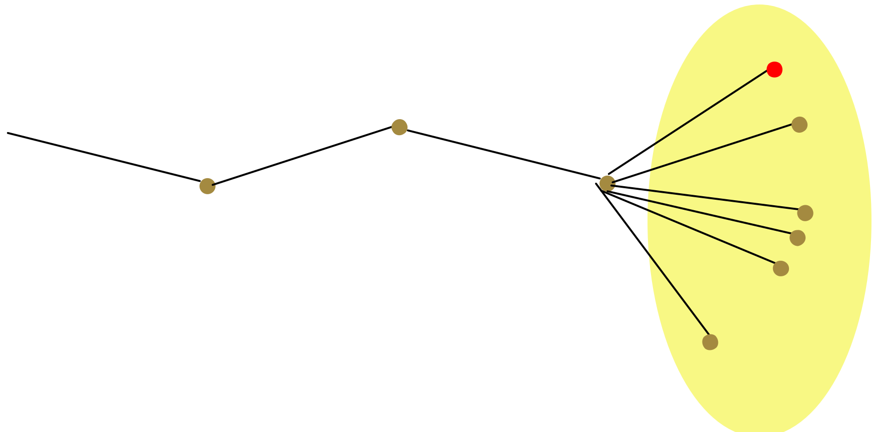

HTTP
Other
To top 10K site
Other
To top 250 data center
Enterprise

IDC topology

- If this is true, we can obtain a list of popular IDCs
- Once we know list of IDCs, can easily use existing tools to find topology
- Then, we could
  - Optimally place distributed computations (such as .NET components and grid computations)
  - Create topology-aware multicast groups
  - Intelligent cache and replicate web content

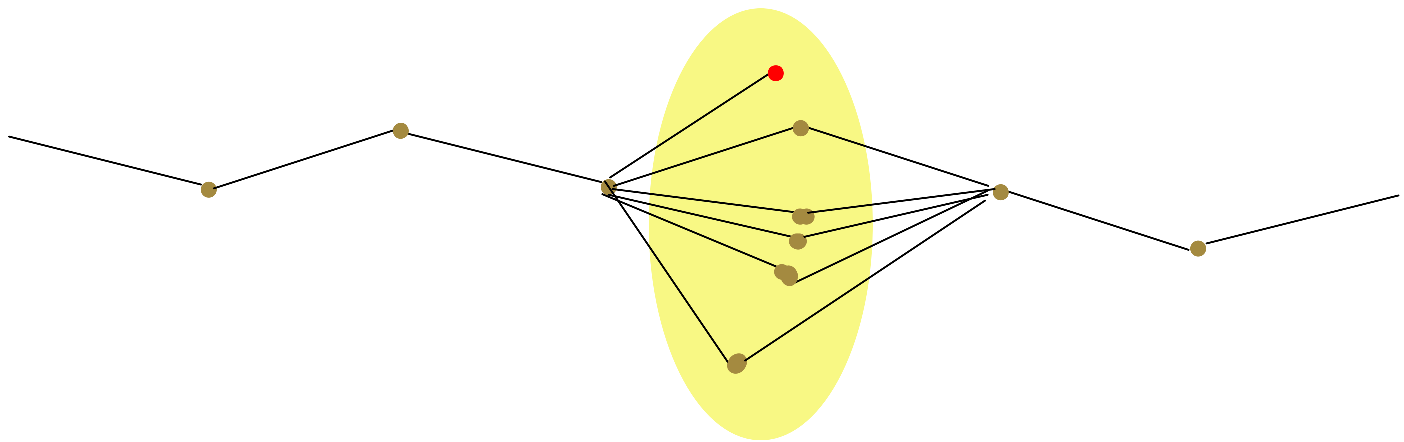
Methodology

- List top websites
- Traceroute to each
- Define *equivalence class*:
  - Set of sites that share the same last hop router
  - Probably all these sites are in the same data center
- But how to distinguish between websites at a data center and a website that is hosted on premises?



Equivalence class refinement

- Step 1: Recompute equivalence classes from multiple vantage points
  - Intersection is the set of websites that share a last hop router from two vantage points
  - Very likely to be hosted at an IDC



Further refinements

- Step 2: Determine ownership of address ranges
  - Further validates ownership
- Step 3: Look at minimum inter-hop delays
  - All websites in the same datacenter will have roughly the same minimum delay from the last hop

Complications

- Router aliasing
  - Same router can report two different IP addresses
- Content distribution networks alias websites
  - Same website shows up in two sites
  - We work around this by tracking (site name, IP)
- Load balancers and firewalls hide sites
  - Need to locate sites using UDP, ICMP, TCP, HTTP
- Datacenters have internal topology
  - Some internal nodes show up in traceroutes, and others don’t
  - Need to massage data to find and correct for this

Preliminary results

- Probed 4320 ‘top’ sites
- Found the last-hop router for 3489 websites
  - For the others, no IP address returned for the last hop
- These fall into 1934 equivalence classes
- Of these, we found 531 IDCs that host 2086 site-tuples
  - Rest (1403 sites) appear to be non-IDC websites
  - In 160 Cities

Summary

- Our hypothesis is that IDC topology concisely represents where the bulk of Internet traffic goes
- If this is true, then it opens the doors for topology-aware computing
- Work is still under way
  - Multiple vantage points
  - Refinement of heuristics

Related work

- Rocketfuel
  - Fast algorithms to determine router topology
- Geotrace
  - Maps routers and servers to geographical locations
- Topology-aware grid computing (UW – Barford)

Conclusions

- Four trends are converging
  - Mobile computers are getting cheaper
  - Batteries last longer
  - Wireless networks are proliferating
  - Internet data centers are aggregating resources
- This motivates four use cases
  - Thin client
  - Global state
  - Coordination
  - Information tagging

Conclusions – contd.

- These use cases have motivated my research agenda on
  - Server virtualization
  - WiFi Roaming
  - IDC topology discovery
- Generally, I'm interested in continuing my research in infrastructure for tetherless computing
  - Choose specific applications for verticals
  - Build out a tetherless community interacting with a datacenter-based computing 'grid'
  - Pose and solve fundamental research problems in this context
    - For example, what does 'fairness' mean in a multi-hop ad hoc network?
  - Bring systems experience to bear to make the 'right' system assumptions

Future research areas

- Look for problems five years out
- Problem selection criteria
  - Relevant
  - Risky
  - High pay off
  - Theoretically sound
  - Synergistic
  - Cross-disciplinary

Specific areas

- Infrastructure for tetherless computing
  - Choose specific applications for verticals
  - Build out a tetherless computing community
  - Pose and solve fundamental research problems in this context
    - For example, what does ‘fairness’ mean in a multi-hop ad hoc network?
- Grid computing

The grand unification!