CS194-24
Advanced Operating Systems Structures and Implementation
Lecture 25

Security (finished)
Quantum Computing
The Swarm

May 5th, 2014

Prof. John Kubiatowicz

http://inst.eecs.berkeley.edu/~cs194-24

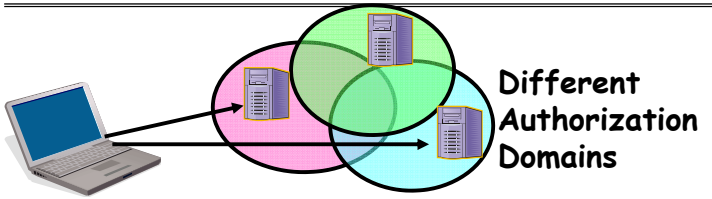
Recall: Mandatory Access Control (MAC)

- Mandatory Access Control (MAC)
  - "A Type of Access control by which the operating system constraints the ability of a *subject* or *initiator* to access or generally perform some sort of operation on an *object* or *target*."
    From Wikipedia
  - Subject: a process or thread
  - Object: files, directories, TCP/UDP ports, etc
  - Security policy is centrally controlled by a security policy administrator: users not allowed to operate outside the policy
  - Examples: SELinux, HiStar, etc.
- Contrast: Discretionary Access Control (DAC)
  - Access restricted based on the identity of subjects and/or groups to which they belong
  - Controls are discretionary – a subject with a certain access permission is capable of passing that permission on to any other subject
  - Standard UNIX model

Recall: Signatures/Certificate Authorities

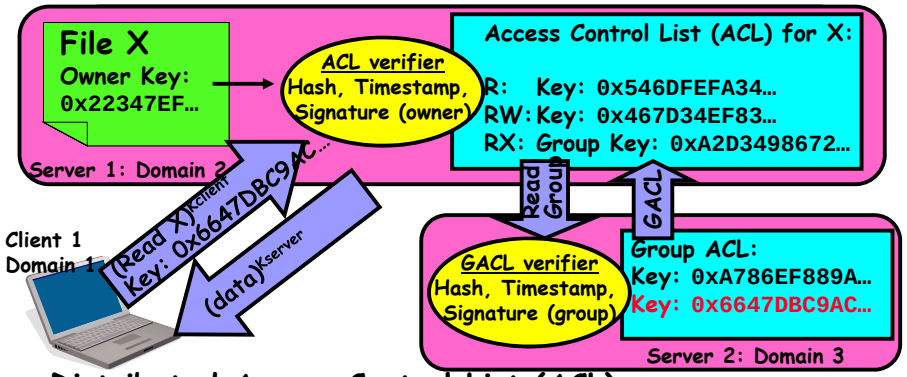
- Can use X_{public} for person X to define their identity
  - Presumably they are the only ones who know $X_{private}$.
  - Often, we think of X_{public} as a "principle" (user)
- Suppose we want X to sign message M?
  - Use private key to encrypt the digest, i.e. $H(M)^{Xprivate}$
  - Send both M and its signature:
    - » Signed message = $[M, H(M)^{Xprivate}]$
  - Now, anyone can verify that M was signed by X
    - » Simply decrypt the digest with X_{public}
    - » Verify that result matches H(M)
- Now: How do we know that the version of X_{public} that we have is really from X???
  - Answer: Certificate Authority
    - » Examples: Verisign, Entrust, Etc.
  - X goes to organization, presents identifying papers
    - » Organization signs X's key: $[X_{public}, H(X_{public})^{CAprivate}]$
    - » Called a "Certificate"
  - Before we use X_{public}, ask X for certificate verifying key
    - » Check that signature over X_{public} produced by trusted authority
- How do we get keys of certificate authority?
  - Compiled into your browser, for instance!

How to perform Authorization for Distributed Systems?



- Issues: Are all user names in world unique?
  - No! They only have small number of characters
    - » kubi@mit.edu → kubitron@lcs.mit.edu → kubitron@cs.berkeley.edu
    - » However, someone thought their friend was kubi@mit.edu and I got very private email intended for someone else…
  - Need something better, more unique to identify person
- Suppose want to connect with any server at any time?
  - Need an account on every machine! (possibly with different user name for each account)
  - OR: Need to use something more universal as identity
    - » Public Keys! (Called "Principles")
    - » People *are* their public keys

Distributed Access Control



- Distributed Access Control List (ACL)
  - Contains list of attributes (Read, Write, Execute, etc) with attached identities (Here, we show public keys)
    - » ACLs signed by owner of file, only changeable by owner
    - » Group lists signed by group key
  - ACLs can be on different servers than data
    - » Signatures allow us to validate them
    - » ACLs could even be stored separately from verifiers

Analysis of Previous Scheme

- Positive Points:
  - Identities checked via signatures and public keys
    - » Client can't generate request for data unless they have private key to go with their public identity
    - » Server won't use ACLs not properly signed by owner of file
  - No problems with multiple domains, since identities designed to be cross-domain (public keys domain neutral)
- Revocation:
  - What if someone steals your private key?
    - » Need to walk through all ACLs with your key and change…!
    - » This is very expensive
  - Better to have unique string identifying you that people place into ACLs
    - » Then, ask Certificate Authority to give you a certificate matching unique string to your current public key
    - » Client Request: (request + unique ID)^{Cprivate}; give server certificate if they ask for it.
    - » Key compromise⇒must distribute "certificate revocation", since can't wait for previous certificate to expire.
  - What if you remove someone from ACL of a given file?
    - » If server caches old ACL, then person retains access!
    - » Here, cache inconsistency leads to security violations!

Analysis Continued

- Who signs the data?
  - Or: How does client know they are getting valid data?
  - Signed by server?
    - » What if server compromised? Should client trust server?
  - Signed by owner of file?
    - » Better, but now only owner can update file!
    - » Pretty inconvenient!
  - Signed by group of servers that accepted latest update?
    - » If must have signatures from all servers ⇒ Safe, but one bad server can prevent update from happening
    - » Instead: ask for a threshold number of signatures
    - » Byzantine agreement (in a moment) can help here
- How do you know that data is up-to-date?
  - Valid signature only means data is valid older version
  - Freshness attack:
    - » Malicious server returns old data instead of recent data
    - » Problem with both ACLs and data
    - » E.g.: you just got a raise, but enemy breaks into a server and prevents payroll from seeing latest version of update
  - Hard problem
    - » Needs to be fixed by invalidating old copies or having a trusted group of servers (Byzantine Agrement?)

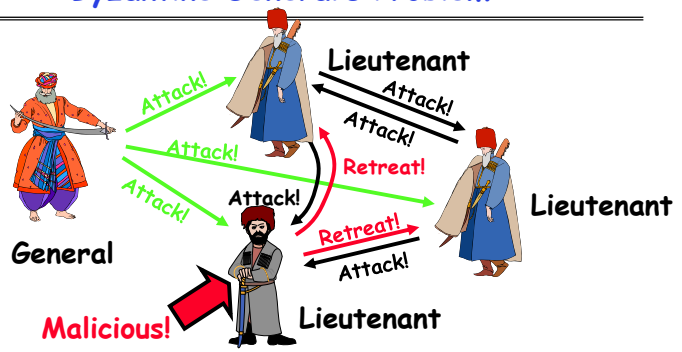
Reprise: Distributed Decision Making

- Why is distributed decision making desirable?
  - Fault Tolerance!
  - Group of machines comes to decision even if one or more fail
    - » Simple failure mode called "failstop" (is this realistic?)
  - After decision made, result recorded in multiple places
- Two-Phase Commit protocol does this
  - Stable log on each machine tracks whether commit has happened
    - » If a machine crashes, when it wakes up it first checks its log to recover state of world at time of crash
  - Prepare Phase:
    - » The global coordinator requests that all participants will promise to commit or rollback the transaction
    - » Participants record promise in log, then acknowledge
    - » If anyone votes to abort, coordinator writes "Abort" in its log and tells everyone to abort; each records "Abort" in log
  - Commit Phase:
    - » After all participants respond that they are prepared, then the coordinator writes "Commit" to its log
    - » Then asks all nodes to commit; they respond with ack
    - » After receive acks, coordinator writes "Got Commit" to log
  - Log helps ensure all machines either commit or don't commit

Distributed Decision Making Discussion (Con't)

- Undesirable feature of Two-Phase Commit: Blocking
  - One machine can be stalled until another site recovers:
    - » Site B writes "prepared to commit" record to its log, sends a "yes" vote to the coordinator (site A) and crashes
    - » Site A crashes
    - » Site B wakes up, check its log, and realizes that it has voted "yes" on the update. It sends a message to site A asking what happened. At this point, B cannot decide to abort, because update may have committed
    - » B is blocked until A comes back
  - A blocked site holds resources (locks on updated items, pages pinned in memory, etc) until learns fate of update
- Alternative: There are alternatives such as "Three Phase Commit" which don't have this blocking problem
- What happens if one or more of the nodes is malicious?
  - Malicious: attempting to compromise the decision making

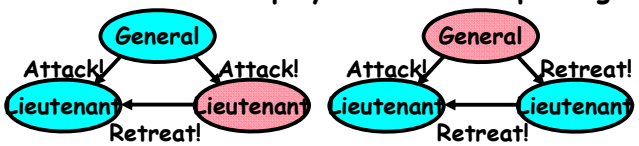
Byzantine General's Problem



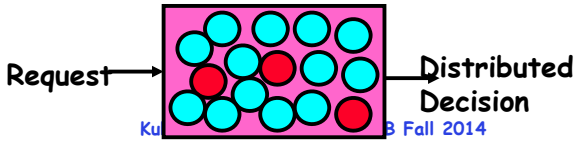
- Byzantine General's Problem (n players):
  - One General
  - n-1 Lieutenants
  - Some number of these (f) can be insane or malicious
- The commanding general must send an order to his n-1 lieutenants such that:
  - IC1: All loyal lieutenants obey the same order
  - IC2: If the commanding general is loyal, then all loyal lieutenants obey the order he sends

Byzantine General's Problem (con't)

- Impossibility Results:
  - Cannot solve Byzantine General's Problem with n=3 because one malicious player can mess up things



  - With f faults, need n > 3f to solve problem
- Various algorithms exist to solve problem
  - Original algorithm has #messages exponential in n
  - Newer algorithms have message complexity $O(n^2)$
    - » One from MIT, for instance (Castro and Liskov, 1999)
- Use of BFT (Byzantine Fault Tolerance) algorithm
  - Allow multiple machines to make a coordinated decision even if some subset of them (< n/3) are malicious



Administrivia

- Don't forget Group evaluations!
  - Please get them in promptly
  - Want all of them by this weekend
    - » Including lab 4: Will post survey sooner rather than later
- Final: Tuesday May 13th
  - 310 Soda Hall
  - 11:30—2:30
  - Bring calculator, 2 pages of hand-written notes
- Review Session
  - Sunday 5/11
  - 4-6PM, 405 Soda Hall
- Final Topics:
  - Focus on second half of term, but still need to be responsible for first half of term material
  - Everything up to last Wednesday!
- Watch for Piazza Survey
  - Try to figure out what we should do with labs
  - What worked, what didn't….

Trusted Computing

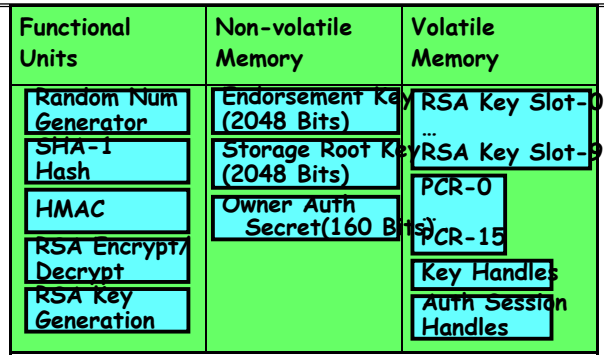
- Problem: Can't trust that software is correct
  - Viruses/Worms install themselves into kernel or system without users knowledge
  - Rootkit: software tools to conceal running processes, files or system data, which helps an intruder maintain access to a system without the user's knowledge
  - How do you know that software won't leak private information or further compromise user's access?
- A solution: What if there were a secure way to validate all software running on system?
  - Idea: Compute a cryptographic hash of BIOS, Kernel, crucial programs, etc.
  - Then, if hashes don't match, know have problem
- Further extension:
  - Secure attestation: ability to *prove* to a remote party that local machine is running correct software
  - Reason: allow remote user to avoid interacting with compromised system
- Challenge: How to do this in an unhackable way
  - Must have hardware components somewhere

TCPA: Trusted Computing Platform Alliance

- Idea: Add a Trusted Platform Module (TPM)
- Founded in 1999: Compaq, HP, IBM, Intel, Microsoft
- Currently more than 200 members
- Changes to platform
  - Extra: Trusted Platform Module (TPM)
  - Software changes: BIOS + OS
- Main properties
  - Secure bootstrap
  - Platform attestation
  - Protected storage
- Microsoft version:
  - Palladium
  - Note quite same: More extensive hardware/software system

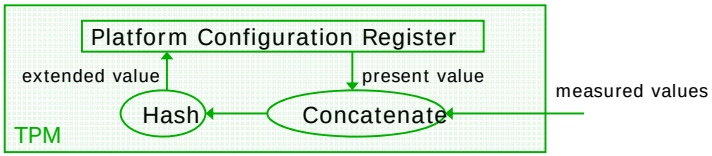
ATMEL TPM Chip
(Used in IBM equipment)

Trusted Platform Module

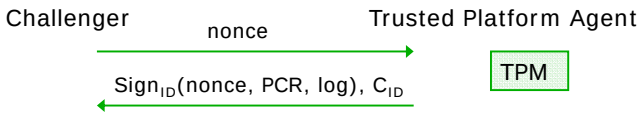


- Cryptographic operations
  - Hashing: SHA-1, HMAC
  - Random number generator
  - Asymmetric key generation: RSA (512, 1024, 2048)
  - Asymmetric encryption/ decryption: RSA
  - *Symmetric encryption/ decryption: DES, 3DES (AES)*
- Tamper resistant (hash and key) storage

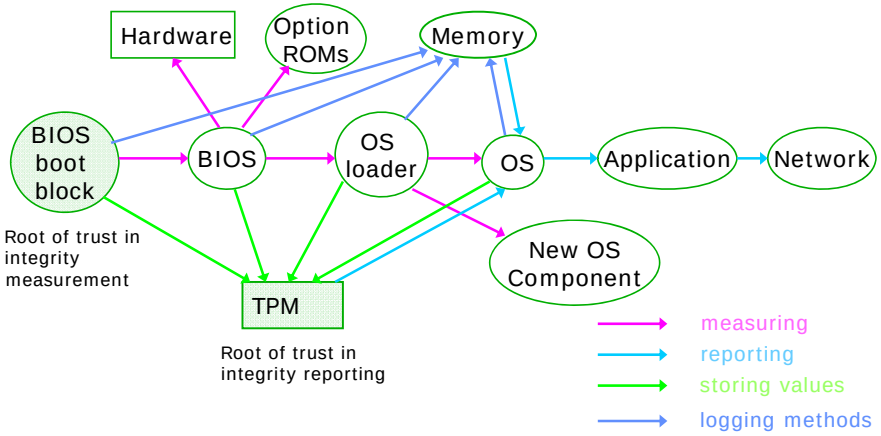
TCPA: PCR Reporting Value



- Platform Configuration Registers (PCR0-16)
  - Reset at boot time to well defined value
  - Only thing that software can do is give new measured value to TPM
    - » TPM takes new value, concatenates with old value, then hashes result together for new PCR
- Measuring involves hashing components of software
- Integrity reporting: report the value of the PCR
  - Challenge-response protocol:

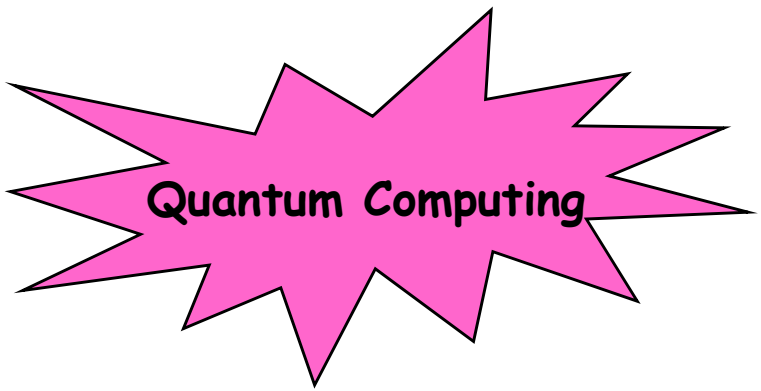


TCPA: Secure bootstrap



Implications of TPM Philosophy?

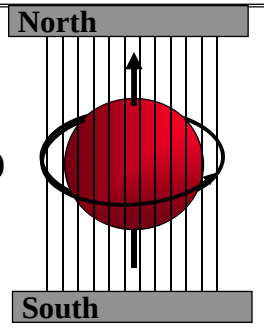
- Could have great benefits
  - Prevent use of malicious software
  - Could help protect privacy of data
- What does "trusted computing" really mean?
  - You are forced to trust hardware to be correct!
  - Could also mean that user is not trusted to install their own software
- Many in the security community have talked about potential abuses
  - These are only theoretical, but very possible
  - Software fixing
    - » What if companies prevent user from accessing their websites with non-Microsoft browser?
    - » Possible to encrypt data and only decrypt if software still matches ⇒ Could prevent display of .doc files except on Microsoft versions of software
  - Digital Rights Management (DRM):
    - » Prevent playing of music/video except on accepted players
    - » Selling of CDs that only play 3 times?



Can we Use Quantum Mechanics to Compute?

- Weird properties of quantum mechanics?
  - Quantization: Only certain values or orbits are good
    - » Remember orbitals from chemistry???
  - Superposition: Schizophrenic physical elements don't quite know whether they are one thing or another
- All existing digital abstractions try to eliminate QM
  - Transistors/Gates designed with classical behavior
  - Binary abstraction: a "1" is a "1" and a "0" is a "0"
- Quantum Computing:
  Use of Quantization and Superposition to compute.
- Interesting results:
  - Shor's algorithm: factors in polynomial time!
  - Grover's algorithm: Finds items in unsorted database in time proportional to square-root of n.

Quantization: Use of "Spin"

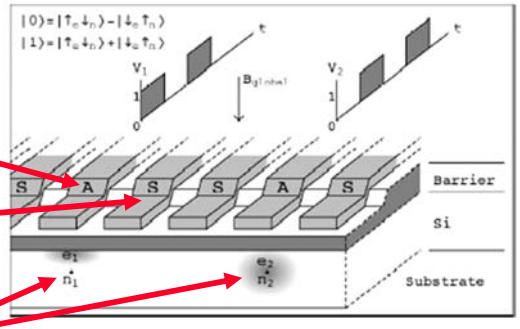


Spin ½ particle:
(Proton/Electron)

Representation:
|0> or |1>

- Particles like Protons have an intrinsic "Spin" when defined with respect to an external magnetic field
- Quantum effect gives "1" and "0":
  - Either spin is "UP" or "DOWN" nothing between

Kane Proposal II (First one didn't quite work)



Single Spin
Control Gates

Inter-bit
Control Gates

Phosphorus
Impurity Atoms

- Bits Represented by combination of proton/electron spin
- Operations performed by manipulating control gates
  - Complex sequences of pulses perform NMR-like operations
- Temperature < 1° Kelvin!

Now add Superposition!

- The bit can be in a combination of "1" and "0":
  - Written as: $\Psi = C_0|0\rangle + C_1|1\rangle$
  - The C's are *complex numbers!*
  - Important Constraint: $|C_0|^2 + |C_1|^2 = 1$
- If *measure* bit to see what looks like,
  - With probability $|C_0|^2$ we will find |0> (say "UP")
  - With probability $|C_1|^2$ we will find |1> (say "DOWN")
- Is this a real effect? Options:
  - This is just statistical – given a large number of protons, a fraction of them ($|C_0|^2$) are "UP" and the rest are down.
  - This is a real effect, and the proton is really both things until you try to look at it
- Reality: second choice!
  - There are experiments to prove it!

Implications: A register can have many values

- Implications of superposition:
  - An *n*-bit register can have 2^n values simultaneously!
  - 3-bit example:

$$\Psi = C_{000}|000\rangle + C_{001}|001\rangle + C_{010}|010\rangle + C_{011}|011\rangle + C_{100}|100\rangle + C_{101}|101\rangle + C_{110}|110\rangle + C_{111}|111\rangle$$

- Probabilities of measuring all bits are set by coefficients:
  - So, prob of getting $|000\rangle$ is $|C_{000}|^2$, etc.
  - Suppose we measure only one bit (first):
    - » We get "0" with probability: $P_0 = |C_{000}|^2 + |C_{001}|^2 + |C_{010}|^2 + |C_{011}|^2$
      Result: $\Psi = (C_{000}|000\rangle + C_{001}|001\rangle + C_{010}|010\rangle + C_{011}|011\rangle)$
    - » We get "1" with probability: $P_1 = |C_{100}|^2 + |C_{101}|^2 + |C_{110}|^2 + |C_{111}|^2$
      Result: $\Psi = (C_{100}|100\rangle + C_{101}|101\rangle + C_{110}|110\rangle + C_{111}|111\rangle)$
- Problem: Don't want environment to *measure* before ready!
  - Solution: Quantum Error Correction Codes!

Spooky action at a distance

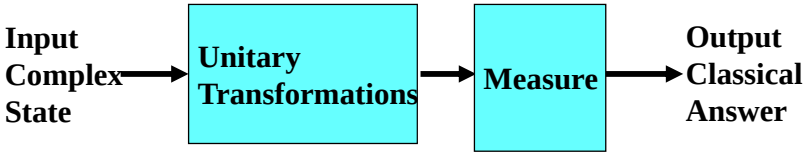
- Consider the following simple 2-bit state:

$$\Psi = C_{00}|00\rangle + C_{11}|11\rangle$$

  - Called an "EPR" pair for "Einstein, Podolsky, Rosen"
- Now, separate the two bits:

- If we measure one of them, it instantaneously sets other one!
  - Einstein called this a "spooky action at a distance"
  - In particular, if we measure a $|0\rangle$ at one side, we get a $|0\rangle$ at the other (and vice versa)
- Teleportation
  - Can "pre-transport" an EPR pair (say bits X and Y)
  - Later to transport bit A from one side to the other we:
    - » Perform operation between A and X, yielding two classical bits
    - » Send the two bits to the other side
    - » Use the two bits to operate on Y
    - » Poof! State of bit A appears in place of Y

Model? Operations on coefficients + measurements



- Basic Computing Paradigm:
  - Input is a register with superposition of many values
    - » Possibly all 2^n inputs equally probable!
  - Unitary transformations compute on coefficients
    - » Must maintain probability property (sum of squares = 1)
    - » Looks like doing computation on all 2^n inputs simultaneously!
  - Output is one result attained by measurement
- If do this poorly, just like probabilistic computation:
  - If 2^n inputs equally probable, may be 2^n outputs equally probable.
  - After measure, like picked random input to classical function!
  - All interesting results have some form of "fourier transform" computation being done in unitary transformation

Security of Factoring

- The Security of RSA Public-key cryptosystems depends on the difficult of factoring a number N=pq (product of two primes)
  - Classical computer: sub-exponential time factoring
  - Quantum computer: polynomial time factoring
- Shor's Factoring Algorithm (for a quantum computer)

Easy 1) Choose random x : $2 \le x \le N-1$.
Easy 2) If $\gcd(x,N) \neq 1$, Bingo!
Hard 3) Find smallest integer r : $x^r \equiv 1 \pmod N$
Easy 4) If r is odd, GOTO 1
Easy 5) If r is even, $a = x^{r/2} \pmod N \Rightarrow (a-1)\times(a+1) = kN$
Easy 6) If $a = N-1$ GOTO 1
Easy 7) ELSE $\gcd(a \pm 1, N)$ is a non trivial factor of N.

Shor's Factoring Algorithm

$$\sum_k |k\rangle|1\rangle \rightarrow \sum_k |k\rangle|x^k\rangle$$

$$= \sum_{w=0}^{r-1} \sum_y |w + ry\rangle|x^w\rangle$$

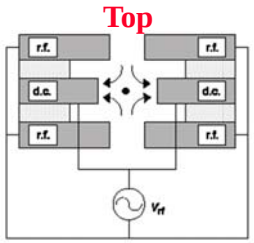
Quantum Fourier Transform

$$\rightarrow \sum_{w=0}^{r-1} \left(\delta_{\frac{0}{r}} \; \delta_{\frac{1}{r}} \; \cdots \; \delta_{\frac{k}{r}} \; \cdots \right) |x^w\rangle$$

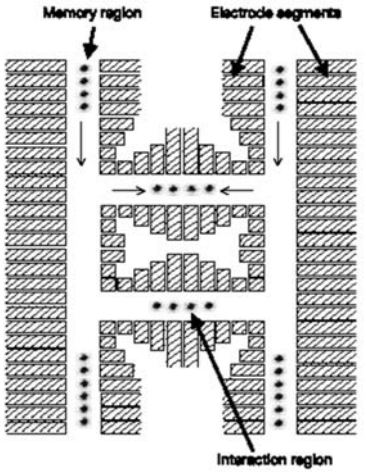
- Finally: Perform measurement
  - Find out r with high probability
  - Get $|y\rangle|a^{w'}\rangle$ where y is of form k/r and w' is related

ION Trap Quantum Computer: Promising technology

Top Cross-Sectional View



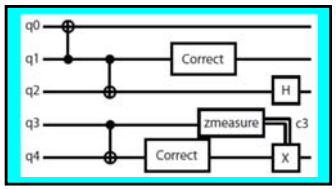
- IONS of Be+ trapped in oscillating quadrature field
  - Internal electronic modes of IONS used for quantum bits
  - MEMs technology
  - Target? 50,000 ions
  - ROOM Temperature!
- Ions moved to interaction regions
  - Ions interactions with one another moderated by lasers



Top View

Proposal: NIST Group

Vision of Quantum Circuit Design



Schematic Capture (Graphical Entry)

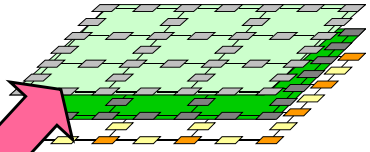
OR

```
cx q1, q0;
cx q1, q2;
correct q1;
h q2;
cx q3, q4;
zmeasure q3, c3;
correct q4;
(@c3==1) x q4;
```

Quantum Assembly (QASM)

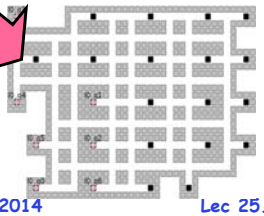
QEC Insertion
Partitioning
Layout
Network Insertion
Error Analysis
...
Optimization

CAD Tool Implementation



Classical Control Teleportation Network

Custom Layout and Scheduling

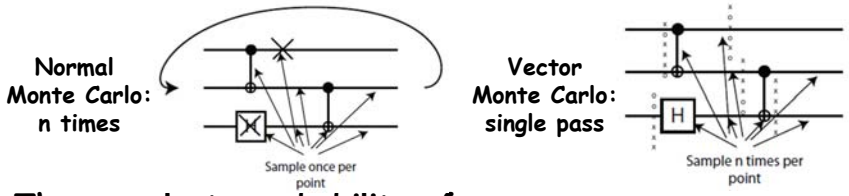


Important Measurement Metrics

- Traditional CAD Metrics:
  - Area
    - » What is the total area of a circuit?
    - » Measured in macroblocks (ultimately μm^2 or similar)
  - Latency ($Latency_{single}$)
    - » What is the total latency to compute circuit *once*
    - » Measured in seconds (or μs)
  - Probability of Success ($P_{success}$)
    - » Not common metric for classical circuits
    - » Account for occurrence of errors and error correction
- Quantum Circuit Metric: ADCR
  - Area-Delay to Correct Result: Probabilistic Area-Delay metric
  - ADCR = Area × E(Latency) $\equiv \frac{Area \times Latency_{single}}{P_{success}}$
  - $ADCR_{optimal}$: Best ADCR over all configurations
- Optimization potential: Equipotential designs
  - Trade Area for lower latency
  - Trade lower probability of success for lower latency

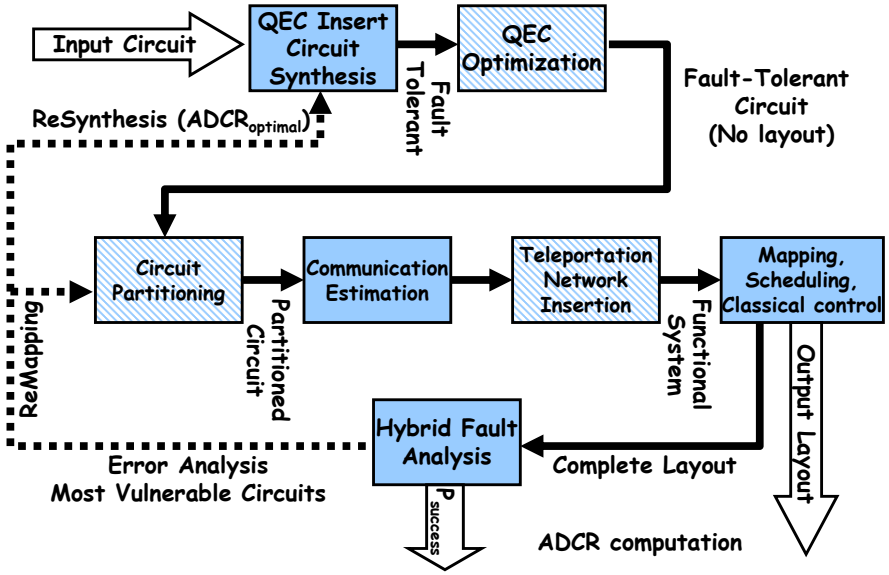
How to evaluate a circuit?

- First, generate a physical instance of circuit
  - Encode the circuit in one or more QEC codes
  - Partition and layout circuit: Dependant of layout heuristics!
    - » Create a physical layout and scheduling of bits
    - » Yields area and communication cost

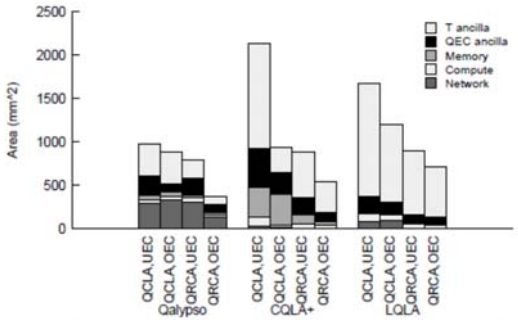


- Then, evaluate probability of success
  - Technique that works well for depolarizing errors: Monte Carlo
    - » Possible error points: Operations, Idle Bits, Communications
  - Vectorized Monte Carlo: n experiments with one pass
  - Need to perform hybrid error analysis for larger circuits
- Finally – Compute ADCR for particular result
  - Repeat as necessary by varying parameters to generate $ADCR_{optimal}$

Quantum CAD flow

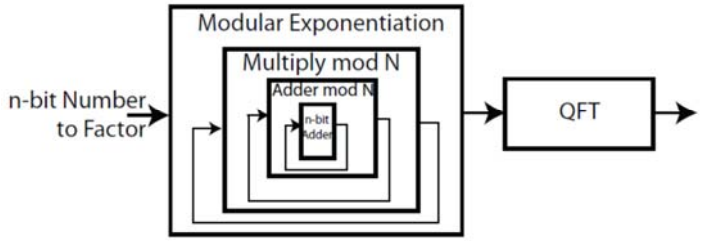


Area Breakdown for Adders



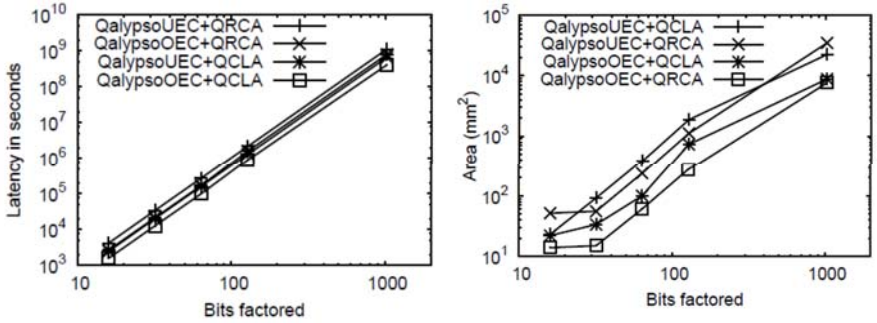
- Error Correction is *not* predominant use of area
  - Only 20-40% of area devoted to QEC ancilla
  - For Optimized Qalypso QCLA, 70% of operations for QEC ancilla generation, but only about 20% of area
- T-Ancilla generation is major component
  - Often overlooked
- Networking is significant portion of area when allowed to optimize for ADCR (30%)
  - CQLA and QLA variants didn't really allow for much flexibility

Investigating 1024-bit Shor's



- Full Layout of all Elements
  - Use of 1024-bit Quantum Adders
  - Optimized error correction
  - Ancilla optimization and Custom Network Layout
- Statistics:
  - Unoptimized version: 1.35×10^{15} operations
  - Optimized Version 1000X smaller
  - QFT is only 1% of total execution time

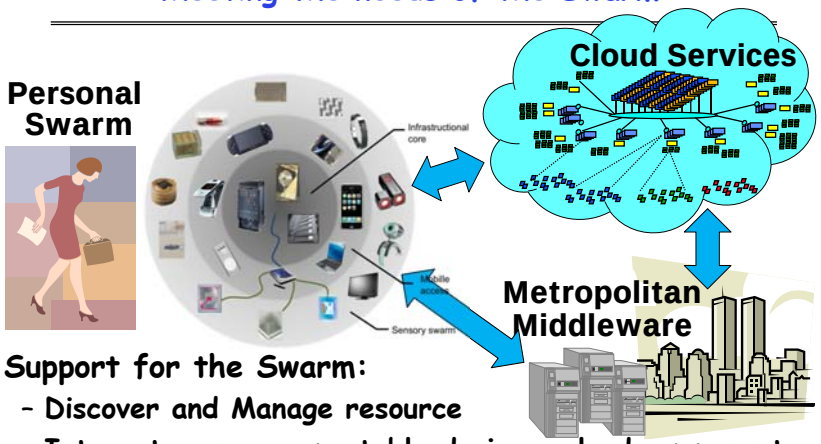
1024-bit Shor's Continued



- Circuits too big to compute $P_{success}$
  - Working on this problem
- Fastest Circuit: 6×10^8 seconds ~ 19 years
  - Speedup by classically computing recursive squares?
- Smallest Circuit: 7659 mm^2
  - Compare to previous *estimate* of 0.9 m^2 = 9×10^5 mm^2

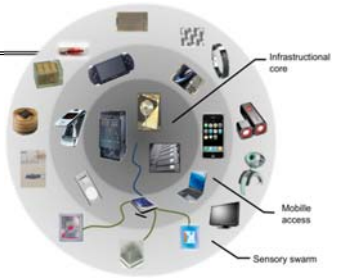
The Swarm

Meeting the needs of the Swarm



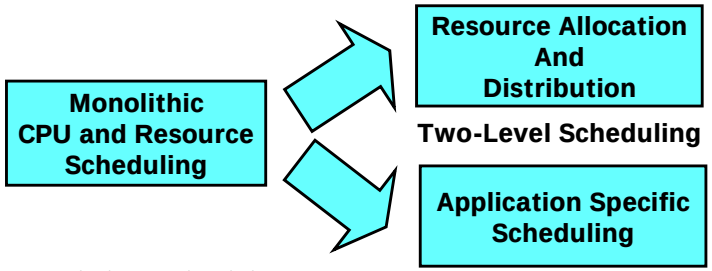
- Support for the Swarm:
  - Discover and Manage resource
  - Integrate sensors, portable devices, cloud components
  - Guarantee responsiveness, real-time behavior, throughput
  - Self-adapt to adjust for failure and provide performance predictability
  - Secure, high-performance, durable, available data

Examples



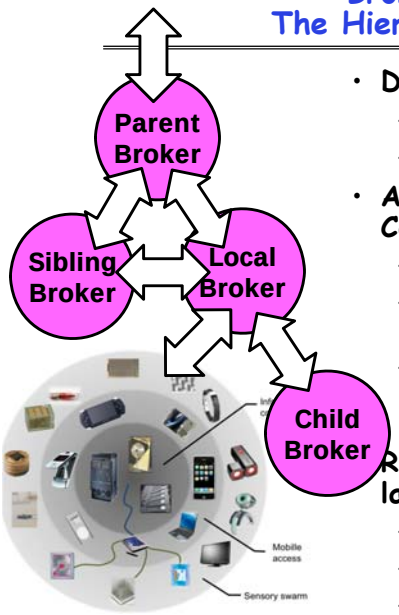
- eWallpaper:
  - Real-Time scheduling of resources
  - Secure loading of code
  - Privacy maintenance of collected information, communication
- Teleconference on nearest wall:
  - Automatic location of resources
    - » Display, Microphone, Camera, Routers
    - » Resources for transcoding, audio transcription
    - » Positional tracking
  - QoS-guaranteed network path to other side
- UnPad:
  - Resource location and allocation
    - » Displays, Microphones, Cameras, etc
    - » High-performance streaming of data from the network
  - ID-Based personalization
    - » RFID, Cellphone connection, other methods for root keys
    - » Targeted advertisement, personalized focus on
  - Deep archival storage ⇒ permanent digital history of activity

Separating Resource Allocation from Resource Usage



- Split monolithic scheduling into two pieces:
  - Course-Grained Resource Allocation and Distribution
    - » Chunks of resources (CPUs, Memory Bandwidth, QoS to Services)
    - » Ultimately a hierarchical process negotiated with service providers
  - Fine-Grained (User-Level) Application-Specific Scheduling
    - » Applications allowed to utilize their resources in any way they see fit
    - » Performance Isolation: Other components of the system cannot interfere with Cells use of resources

Brokering Service: The Hierarchy of Ownership

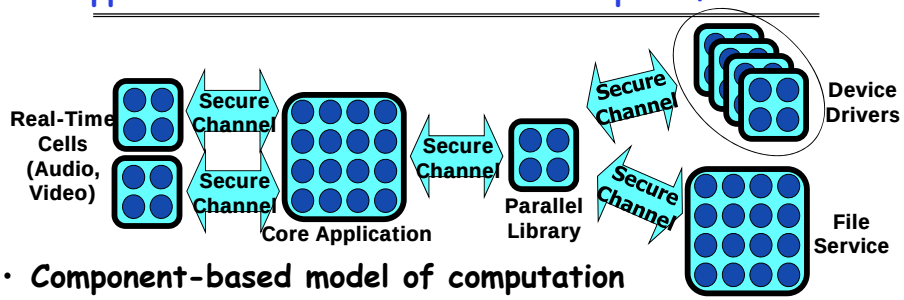


- Discover Resources in "Domain"
  - Devices, Services, Other Brokers
  - Constraints: Ownership, Access Control
- Allocate and Distribute Resources to Components that need them
  - Dynamically optimize execution
  - Hand out Service-Level Agreements (SLAs) to Software Components
  - Deny admission to application components when violate existing agreements
- Resources described via declarative language: properties + requirements
  - Model of cyber-physical interactions
  - Requirements for usage
  - Constraints placed on other resources

Resource Container: the Cell

- Properties of a Cell
  - A user-level software component with guaranteed resources
  - Has full control over resources it owns ("Bare Metal")
  - Contains at least one memory protection domain (possibly more)
  - Contains a set of secured channel endpoints to other Cells
  - Hardware-enforced security context to protect the privacy of information and decrypt information (a Hardware TCB)
- Each Cell schedules its resources exclusively with application-specific user-level schedulers
  - Gang-scheduled hardware thread resources ("Harts")
  - Virtual Memory mapping and paging
  - Storage and Communication resources
    - » Cache partitions, memory bandwidth, power or energy
  - Use of Guaranteed fractions of system services

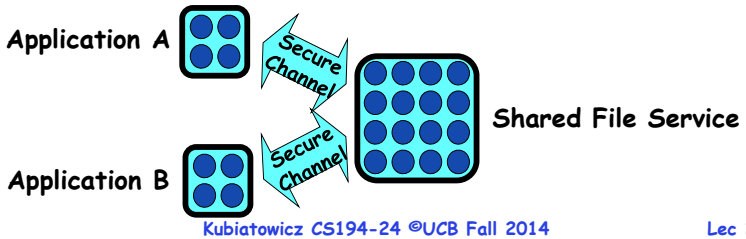
Applications are Interconnected Graphs of Services



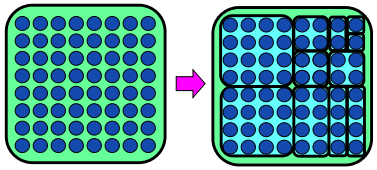
- Component-based model of computation
  - Applications consist of interacting components
  - Explicitly asynchronous/non-blocking
  - Components may be local or remote
- Channel Interface ⇒ Service API, Security Boundary
  - Channels are points at which data may be compromised
  - Channels define points for QoS constraints
  - Fault tolerance and adaptation by evolving connections

Impact on the Programmer

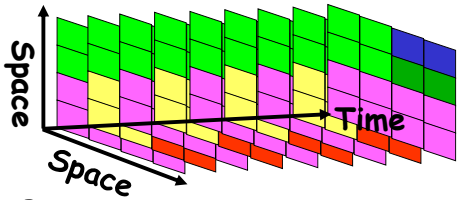
- Connected graph of Cells ⇔ Object-Oriented Programming
  - Lowest-Impact: Wrap a functional interface around channel
    - » Cells hold "Objects", Secure channels carry RPCs for "method calls"
  - Greater Parallelism: Event triggered programming
- Applications compiled from abstract graph description
  - Independent of location or identity of services
- Shared services complicate resource isolation:
  - How to ensure each client gets well-defined fraction of service?
  - Distributed resource attribution (application as distributed graph)



Space-Time Partitioning ⇒ Cell

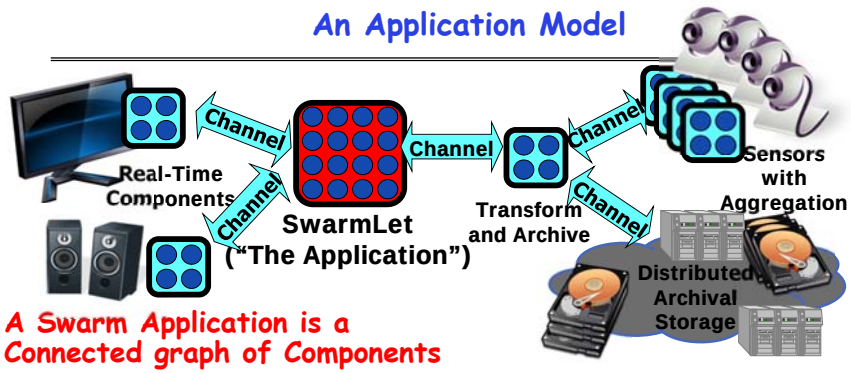


- Spatial Partition: Performance isolation
  - Each partition receives a vector of basic resources
    - » A number HW threads
    - » Chunk of physical memory
    - » A portion of shared cache
    - » A fraction of memory BW
    - » Shared fractions of services
  - Use of Virtualization extensions (e.g. Vt-x)



- Partitioning varies over time
  - Fine-grained multiplexing and guarantee of resources
    - » Resources are gang-scheduled
- Controlled multiplexing, not uncontrolled virtualization
- Partitioning adapted to the system's needs

An Application Model



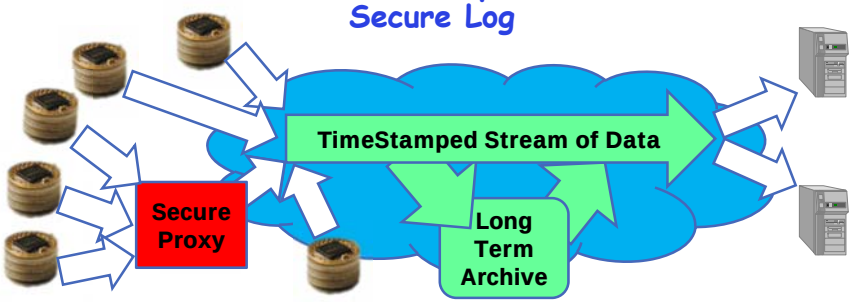
- A Swarm Application is a Connected graph of Components
  - Globally distributed, but locality and QoS aware
  - Avoid Stovepipe solutions through reusability
- Many components are *Shared Services* written by programmers with a variety of skill-sets and motivations
  - Well-defined semantics and a managed software version scheme
  - Service Level Agreements (SLA) with micropayments
- Many are "Swarmlets" written by *domain programmers*
  - They care *what* application does, not *how* it does it

DataCentric Vision

- Hardware resources are a commodity
  - Computation resource fails? Get another
  - Sensor fails? Find another
  - Change your location? Find new resources
- All that really matters is the information
  - Integrity, Privacy, Availability, Durability
  - Hardware to prevent accidental information leakage
- We need a new Internet for the Internet of Things
  - Communication and Storage are really duals
  - Why separate them?

The Universal Data Plane

Location-Independent Secure Log

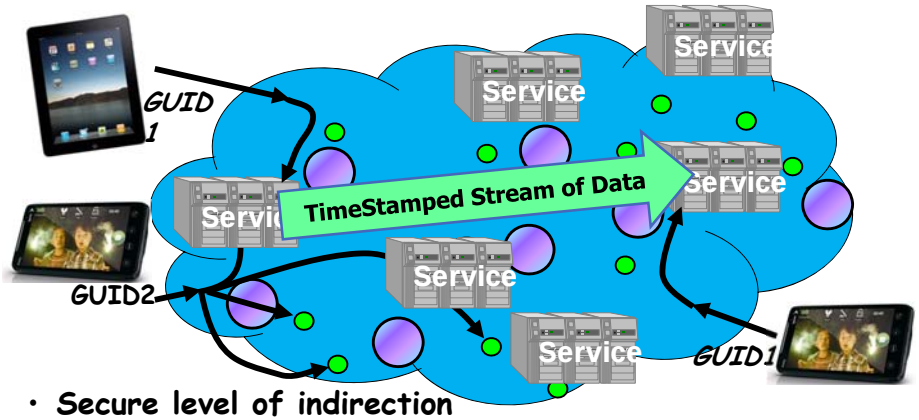


- Multi-Source, Multi-Sink Secure Log
  - Append-only with timestamps and QoS specifications
  - Only authorized writers can append information
  - Only authorized readers can read data (random access)
- Low-functionality sensors can operate through proxy
  - Sign, Encrypt, Route – proxies can be deployed with sensors
- Transparent, Long-term Archival Storage
  - Support for Transactions, Indexing, Random Access

Log-Structured Communication *Is* the New Universal Primitive!

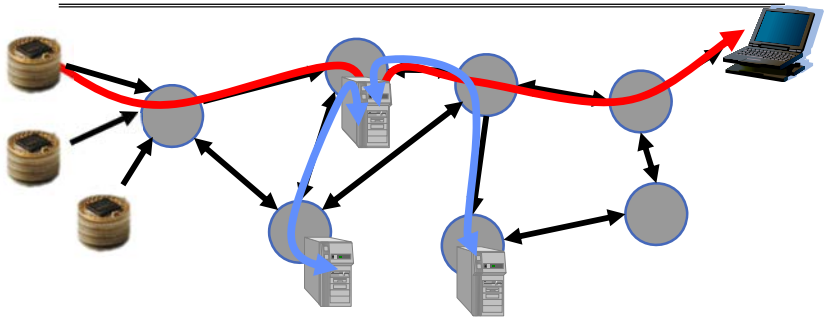
- Location-Independent routing to Logs *by Identifier*
  - Identifiers Represent Sinks and/or Sources of Data and are provided by discovery mechanism
  - Routing optimized by Swarm-Routing Stack (SRS) (adapts to failure and topology changes)
- QoS specifications with respect to latency and/or BW
  - TimeStamps used for "deliver-by" scheduling (i.e. pTides)
  - QoS attached to sources and/or sinks (state in network)
- Data Commitment and Archiving performed securely
  - No single node can prevent or authorize changes
  - Data encrypted and users authenticated at all times
  - Data reordered by timestamps (subject to constraints)

Peer-to-Peer overlays for Object Location and Routing



- Secure level of indirection
  - Automatic adaptation to underlying topology
- Commitment leaders and/or elements at tail of log found via automatic routing

QoS Routing Constraints are Integral to Model

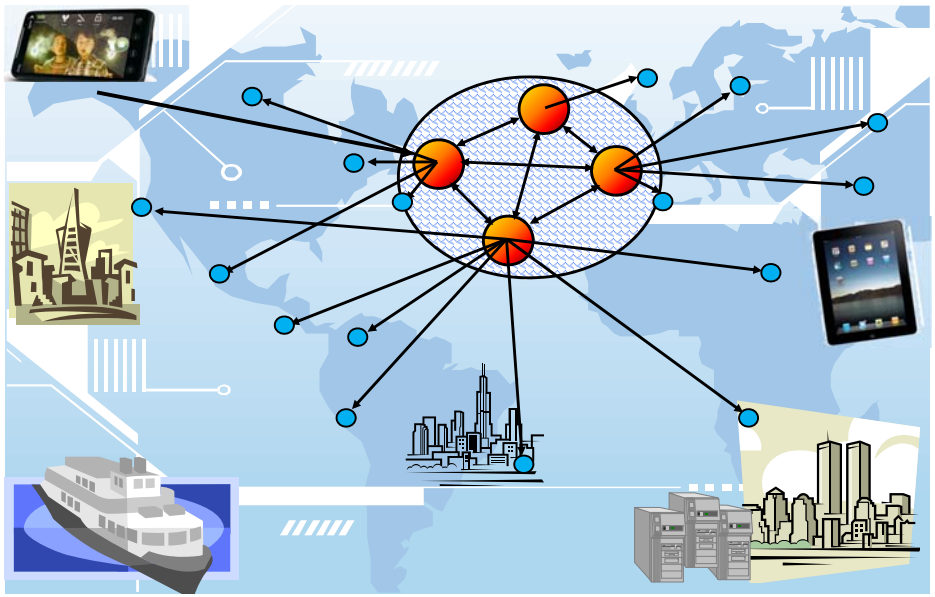


- Physical Routing ⇒ Overlay Networks
  - Aware of topology and time of flight through nodes
  - Can Utilize physical QoS when available (i.e AVB)
- Location-independent Routing through Overlay
  - Including Services (Log commit) Along Path
- Collaborative functionality runs on through overlay
  - Byzantine Commitment/Archiving

Time-Based Log Use Cases

1. Short-term archived stream of information
   - Authenticated channel of data bits ordered by timestamp
   - Data expires after specified time period
2. Log-structured storage
   - Infrastructure provides associative lookup from log
     - » Older entries read directly from the underlying infrastructure
   - Easy to provide object storage via copy-on-write
3. Externally Consistent Transactions
   - Commitment order based on timestamps
   - Automatic mechanism for aborting transactions that would violate serialization (i.e. Spanner)
   - Multi-stream transactions possible

Online Deep Archival Storage (Using Erasure Codes such as Reed-Solomon)



Trusted Swarm Platform

- External Data Encrypted All The Time
- Only decrypted in "Data Jails" (trusted platform)
  - Build in hardware or in software with secure attestation
  - Data leaving cell automatically reencrypted
- Trusted Platform given keys to do its work
  - Keys never given out to application software
- Similar idea: Hardware micropayment support

The Internet for the IoT

- Duality between communication and storage
  - Why explicitly distinguish them?
  - The "Data Grid" equivalent to the "Power Grid"
  - All data is *read-only* and time stamped when it enters the grid and preserved as long as it stays in the grid
- Location Independent routing provides convenient glue for constructing Swarm applications
  - Endpoints can be services, sensors, or archival objects
  - Automatically locate close objects with given endpoint
- Dynamic Optimization: Gain advantages normally available only to large internet providers
  - Generate optimized multicast networks when necessary
  - Construct content distribution networks (CDNs) on the fly
- Security, authentication, privacy, micropayments

Conclusion (1/2)

- Distributed identity
  - Use cryptography (Public Key, Signed by PKI)
- Distributed storage example
  - Revocation: How to remove permissions from someone?
  - Integrity: How to know whether data is valid
  - Freshness: How to know whether data is recent
- Byzantine General's Problem: distributed decision making with malicious failures
  - One general, n-1 lieutenants: some number of them may be malicious (often "f" of them)
  - All non-malicious lieutenants must come to same decision
  - If general not malicious, lieutenants must follow general
  - Only solvable if $n \geq 3f+1$
- Trusted Hardware
  - A secure layer of hardware that can:
    - » Generate proofs about software running on the machine
    - » Allow secure access to information without revealing keys to (potentially) compromised layers of software
  - Cannonical example: TPM

Conclusion (2/2)

- Quantum Computing
  - Using interesting properties of physics to compute
  - Noise is one of the most complex aspects
  - At a stage where Computer Aided Design (CAD) makes sense
  - Quantum Circuit Metric: ADCR
    - » Area-Delay to Correct Result: Probabilistic Area-Delay metric
    - » ADCR = Area × E(Latency)
      $ADCR_{optimal}$: Best ADCR over all configurations
- Services for the Swarm:
  - Use of Resources negotiated hierarchically
  - Underlying Execution environment guarantees QoS
  - New Resources constructed from Old ones:
    - » Aggregate resources in combination with QoS-Aware Scheduler
    - » Result is a *new* resource that can be negotiated for
  - Continual adaptation and optimization
- Let's give a hand to Palmer and Vedant – the labs wouldn't exist without them!