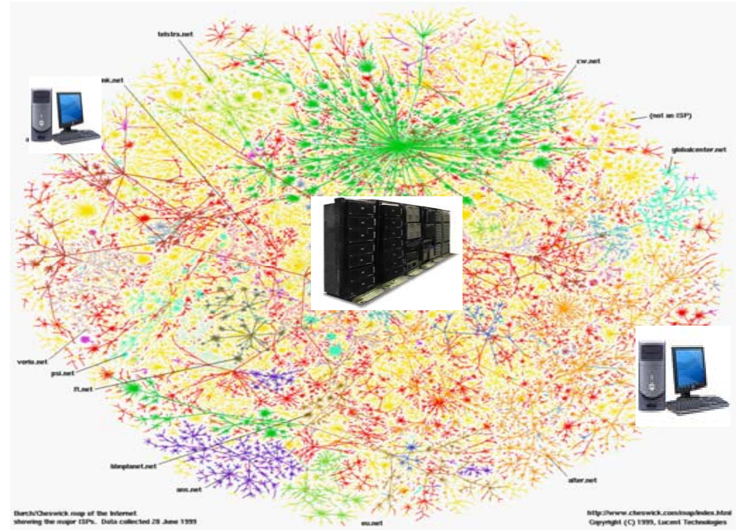
# CS162
# Operating Systems and Systems Programming
# Lecture 25

## Extra Topics:
## IoT, Quantum Computing

May 4th, 2015

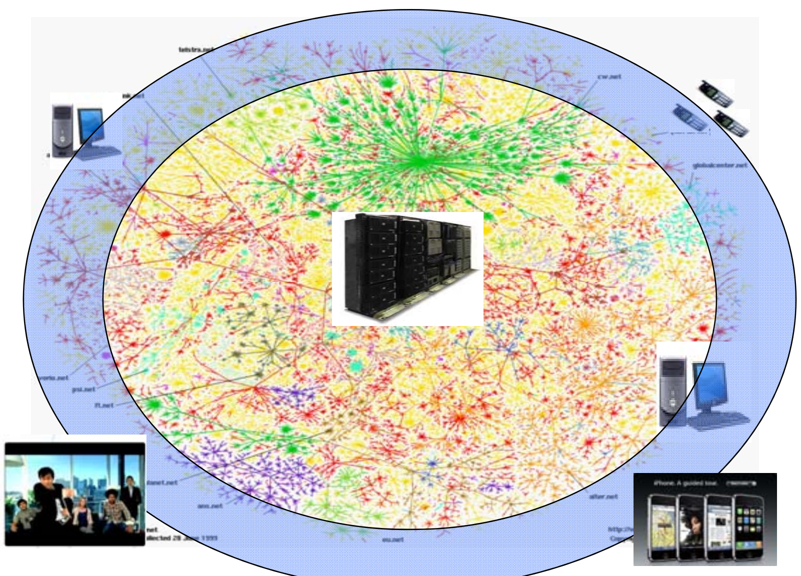
Prof. John Kubiatowicz
http://cs162.eecs.Berkeley.edu

## 1997 - The Internet of Every Computer



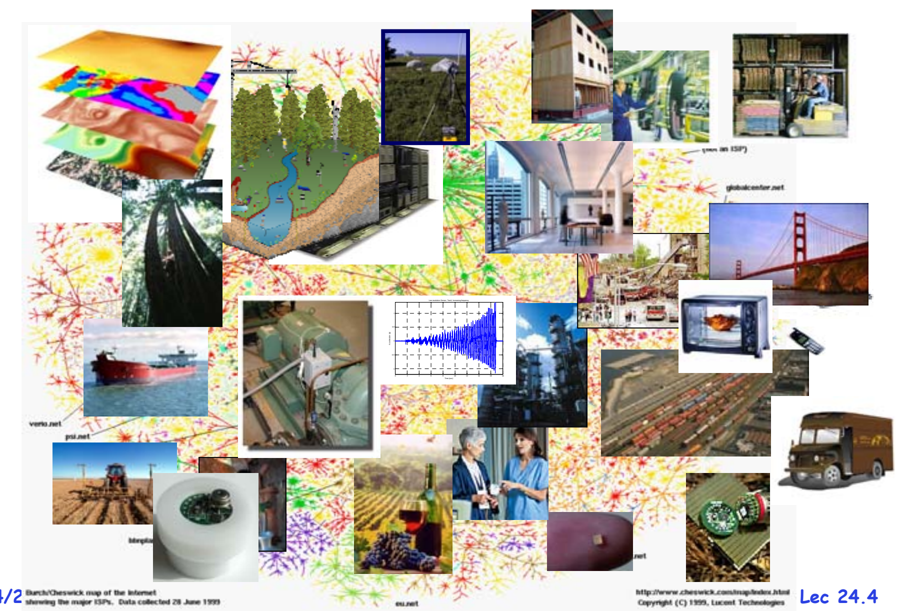


## 2007 - The Internet of Every Body





## 2017 - The Internet of Everyday Things

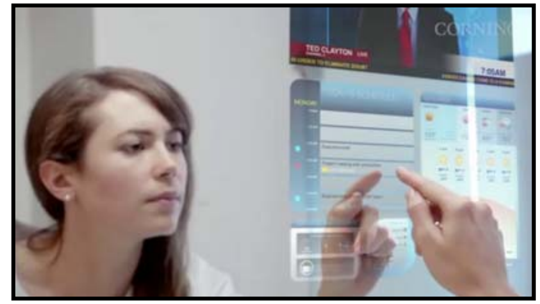


## Why "Real" Information is so Important




## Resources in a Smart Space (2011)

- Potential Displays Everywhere
  - Walls, Tables, Appliances, Smart Phones, Google Glasses….
- Audio Output Everywhere
- Inputs Everywhere
  - Touch Surfaces
  - Cameras/ Gesture Tracking
  - Voice
- Context Tracking
  - Who is Where
  - What do they want
  - Which Inputs map to which applications

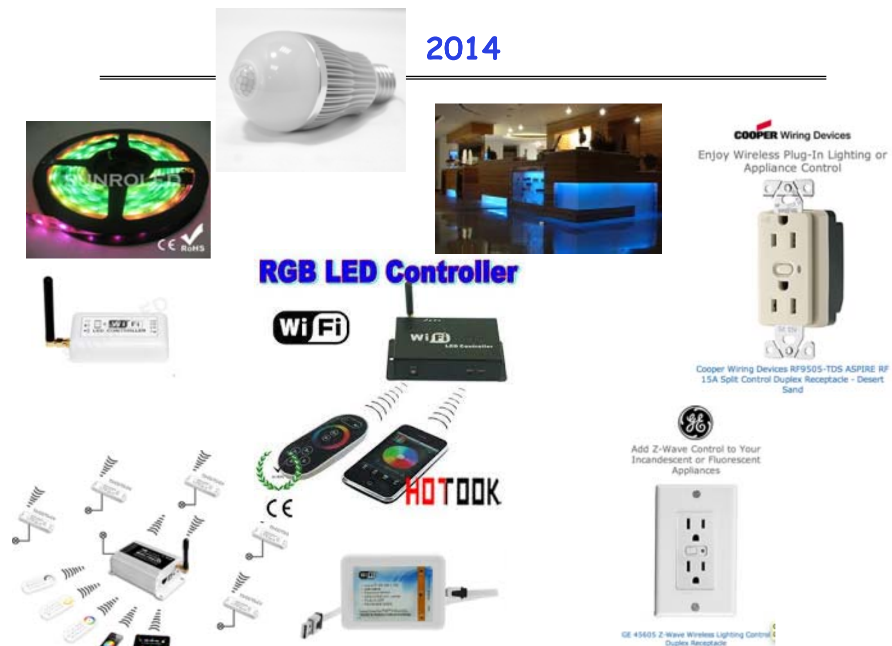




## 2013




## 2014



# 2013


The Nest makes headlines!



# 2014




# 2014

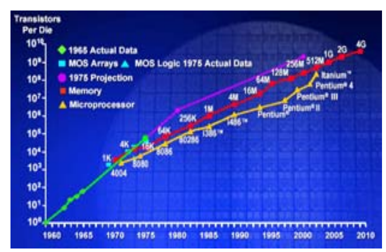




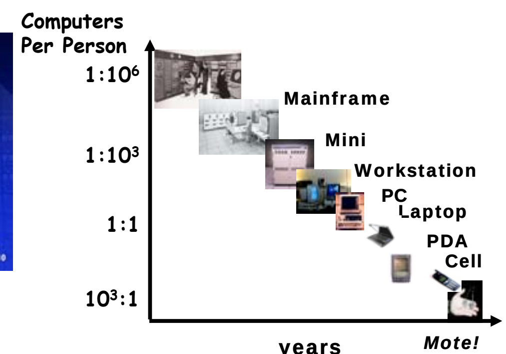
# Broad Technology Trends

**Moore's Law:** # transistors on cost-effective chip doubles every 18 months

**Bell's Law:** a new computer class emerges every 10 years




Today: 1 million transistors per $




Same fabrication technology provides CMOS radios for communication and micro-sensors

## 'Low-Tech' Enabling Technology

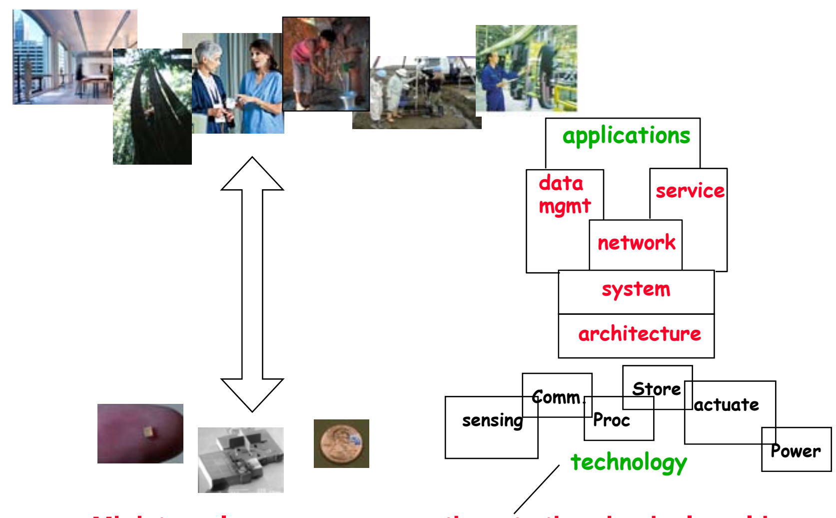

Microcontroller Flash Storage Radio Communication Sensors

IEEE 802.15.4



## The Systems Challenge

Monitoring & Managing Spaces and Things




Miniature, low-power connections to the physical world

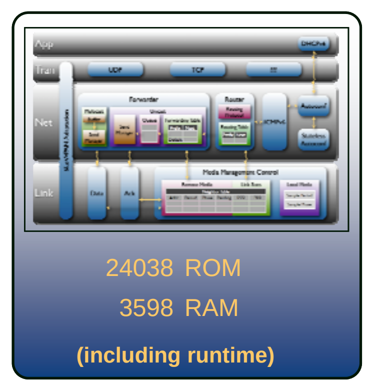


## Key WSN Research Developments

- Event-Driven Component-Base Operating System
  - Framework for building System & Network abstractions
  - Low-Power Protocols
  - Hardware and Application Specific
- Idle listening
  - All the energy is consumed by listening for a packet to receive
  - => Turn radio on only when there is something to hear
- Reliable routing on Low-Power & Lossy Links
  - Power, Range, Obstructions => multi-hop
  - Always at edge of SNR => loss is common
  - => monitoring, retransmission, and local rerouting
- Trickle – don't flood (tx rate < 1/density, and < info change)
  - Connectivity is determined by physical points of interest, not network designer.
  - never naively respond to a broadcast
  - re-broadcast very very politely



## Internet of Every Thing – Realized 2008




* Production implementation on TI msp430/cc2420

| | ROM | RAM |
|---|---|---|
| CC2420 Driver | 3149 | 272 |
| 802.15.4 Encryption | 1194 | 101 |
| Media Access Control | 330 | 9 |
| Media Management Control | 1348 | 20 |
| 6LoWPAN + IPv6 | 2550 | 0 |
| Checksums | 134 | 0 |
| SLAAC | 216 | 32 |
| DHCPv6 Client | 212 | 3 |
| DHCPv6 Proxy | 104 | 2 |
| ICMPv6 | 522 | 0 |
| Unicast Forwarder | 1158 | 451 |
| Multicast Forwarder | 352 | 4 |
| Message Buffers | 0 | 2048 |
| Router | 2050 | 106 |
| UDP | 450 | 6 |
| TCP | 1674 | 50 |

- Footprint, power, packet size, & bandwidth
- Open version 27k / 4.6k

ARCHROCK

# Internet of Every Thing – standardized 2010

```
ROLL                                                   T. Winter, Ed.
Internet-Draft
Intended status: Standards Track                       P. Thubert, Ed.
Expires: April 4, 2011                                  Cisco Systems
                                                            A. Brandt
                                                        Sigma Designs
                                                           T. Clausen
                                            LIX, Ecole Polytechnique
                                                               J. Hui
                                              Arch Rock Corporation
                                                            R. Kelsey
                                                  Ember Corporation
                                                             P. Levis
                                                 Stanford University
                                                            K. Pister
                                                       Dust Networks
                                                            R. Struik

                                                          JP. Vasseur
                                                        Cisco Systems
                                                      October 1, 2010

      RPL: IPv6 Routing Protocol for Low power and Lossy Networks
                      draft-ietf-roll-rpl-12

Abstract

   Low power and Lossy Networks (LLNs) are a class of network in which
   both the routers and their interconnect are constrained.  LLN routers
```

```
2008-02-15 charter

Routing Over Low power and Lossy networks (roll)
------------------------------------------------

 Charter

 Current Status: Active Working Group

 Chair(s):
     JP Vasseur  <jpv@cisco.com>
     David Culler  <culler@eecs.berkeley.edu>
```

IPSO Alliance – Enabling the INTERNET OF THINGS

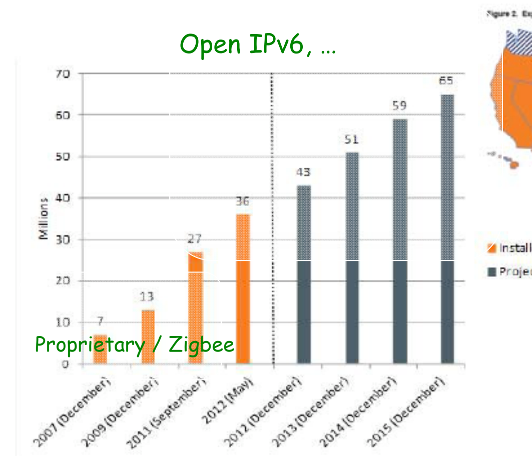
ZigBee Alliance

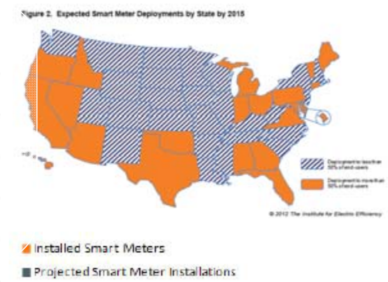
ZigBee Smart Energy Version 2.0 Documents

ZigBee Smart Energy version 2.0 will be IP-based and offer a variety of new features.



# Smart meter rollouts



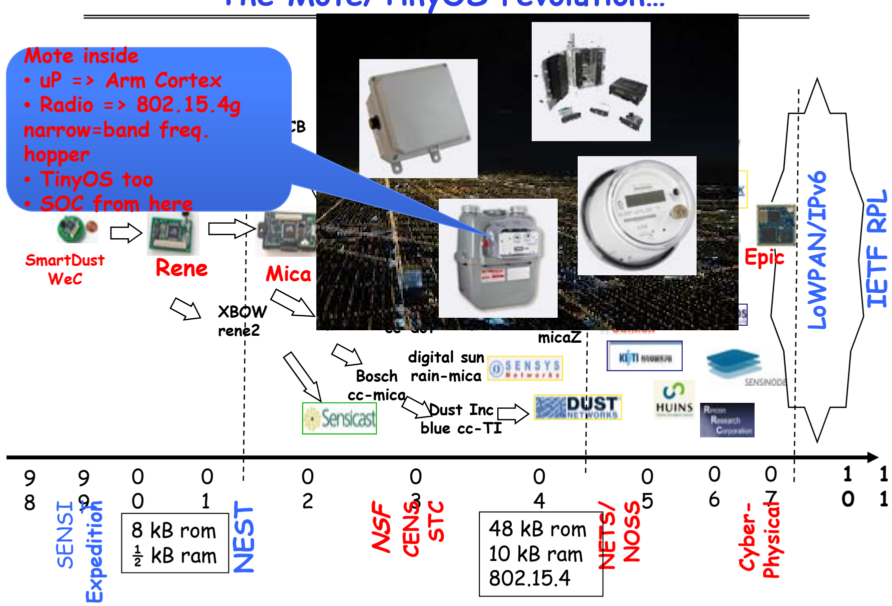


Source: Institute for Electric Efficiency, Federal Energy Regulatory Commission[2]

http://www.edisonfoundation.net/iee/Documents/IEE_SmartMeterRollouts_0512.pdf

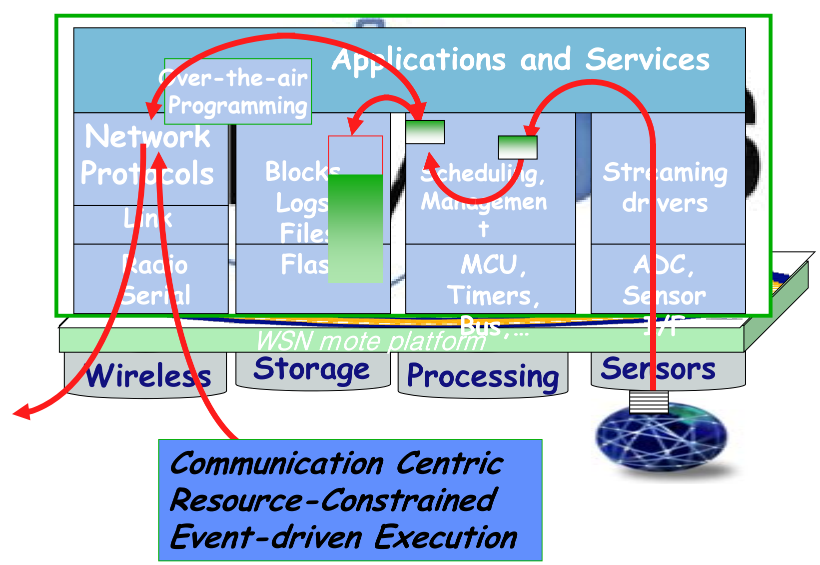


# The Mote/TinyOS revolution…

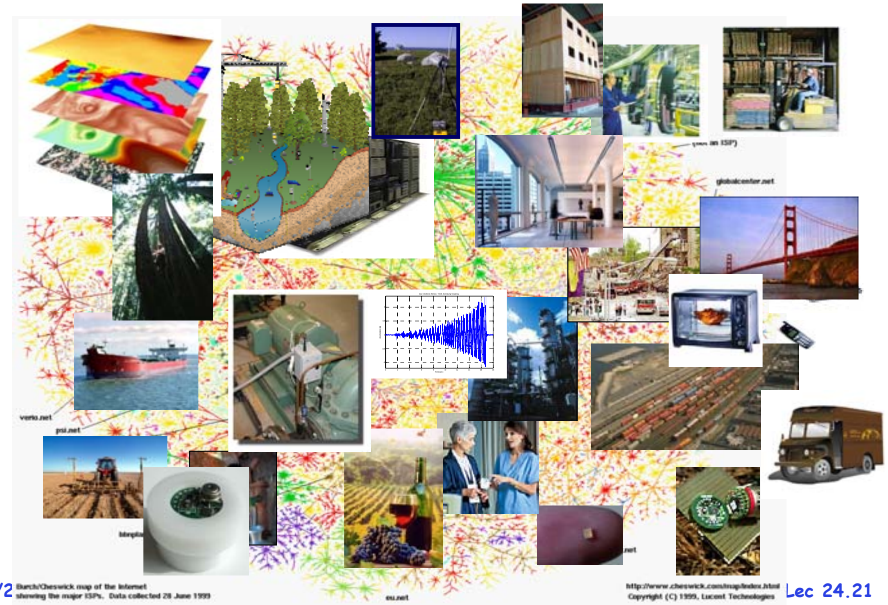





# TinyOS – Framework for Innovation

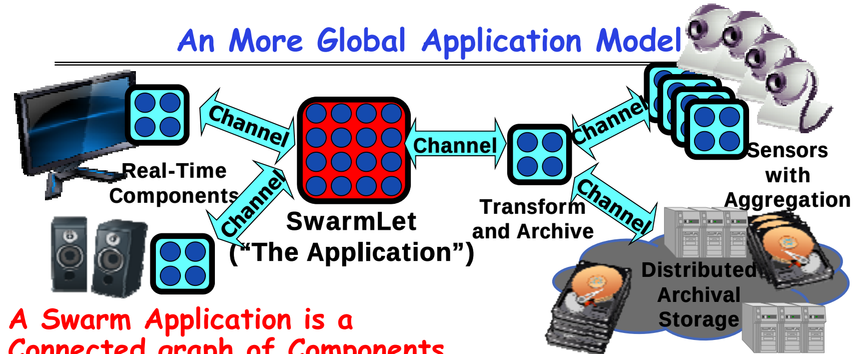


## Back to The Internet of Everything...?





## An More Global 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

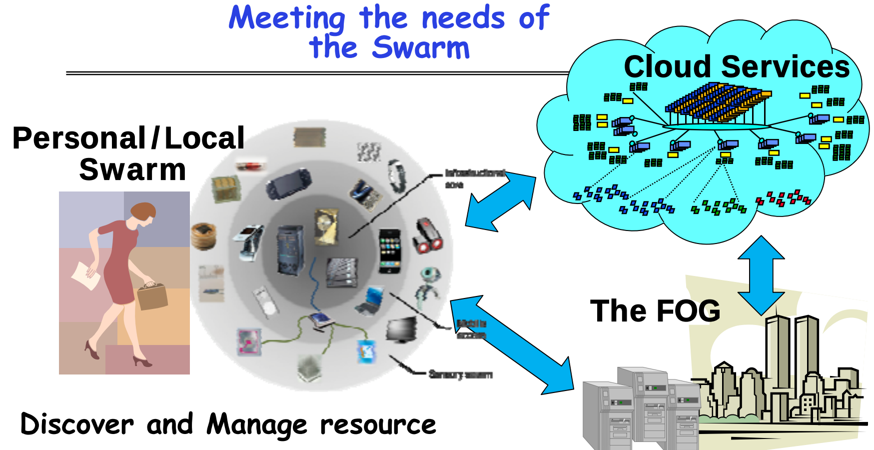


## SWARMLETs

- SWARMLET: a software component written by domain programmer that is easy to write but exhibits sophisticated behavior by exploiting services distributed within the infrastructure
- Swarmlets specify their needs in terms of human-understandable requirements
  - Necessary Services, Frame rates, Minimum Bandwidths
  - Locality, Ownership, and Micropayment parameters for sensors and/or data
- Swarmlets may evolve into Shared Services
- Programmers of Services used by Swarmlets think in terms of contracts provided to Swarmlets



## Meeting the needs of the Swarm



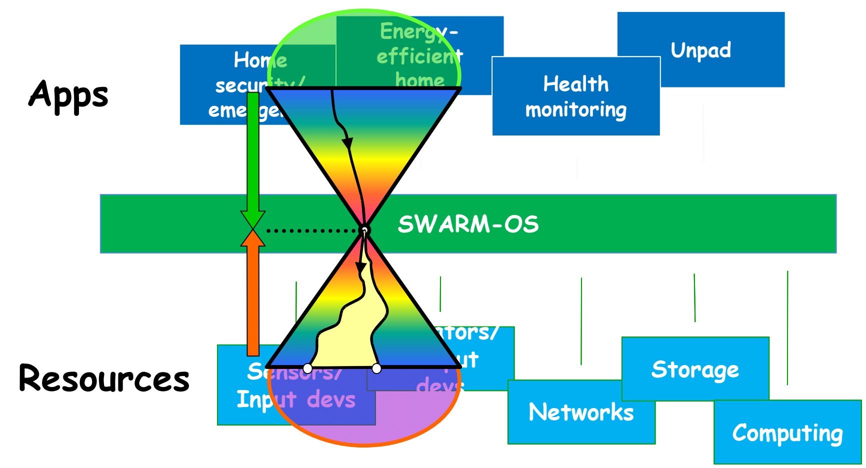

- Discover and Manage resource
- Integrate sensors, portable devices, cloud components
- Guarantee responsiveness, real-time behavior, throughput
- Self-adapt to failure and provide performance predictability
- Secure, high-performance, durable, available information
- Monetize resources when necessary: micropayments

## Support for Swarm Applications

- Resource Discovery
  - Which resources /services are available?
  - What are these resources and what are their capabilities?
  - Who owns them and how much do I need?
- Real Time Requirements
  - Sophisticated multimedia interactions
  - Control of/interaction with health-related devices
- Responsiveness Requirements
  - Provide a good interactive experience to users
- Explicitly Parallel Components
  - Components exploit parallelism when possible
- Direct Interaction with Cloud storage and computation
  - Potentially extensive use of remote services
  - Serious security/data vulnerability concerns



## The Missing Link?




SWARM-OS: A mediation layer that discovers resources and connects them with applications



## What about the "FOG" and "Cloud"? New Abstraction: the Cell

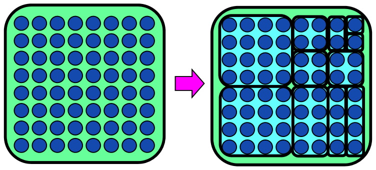
- Properties of a Cell: Service Level Guarantees
  - A user-level software component with guaranteed resources
  - Has full control over resources it owns ("Bare Metal")
  - Contains at least one memory protection domain (possibly more)
  - Contains a set of secured channel endpoints to other Cells
  - Contains a security context which may protect and decrypt information
- When mapped to the hardware, a cell gets:
  - Gang-schedule hardware thread resources ("Harts")
  - Guaranteed fractions of other physical resources
    - » Physical Pages (DRAM), Cache partitions, memory bandwidth, power
  - Guaranteed fractions of system services
- Predictability of performance ⇒
  - Ability to model performance vs resources
  - Ability for user-level schedulers to better provide QoS

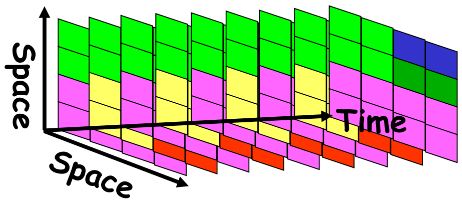


## Cell Implementation Platform: Tessellation Version 2

- Tessellation Operating System
  - Provides basic Cell Implementation
  - Build on the Xen Hypervisor
- Why Xen?
  - Provides clean starting point for resource containers
  - Leverage mature OS (Linux) device support, critical drivers can be isolated in a stub domain
  - Framework for developing VM schedulers
  - Mini-OS, a lightweight POSIX-compatible Xen guest OS, is basis for the customizable app runtime
  - Support for ARM and x86
- Unikernels: Software Appliances
  - Small compiled kernels with only enough components to support one application
  - Every component has its own resource container
- Dynamic resource optimization framework

## Implementing Cells: Space-Time Partitioning






- 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
- 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

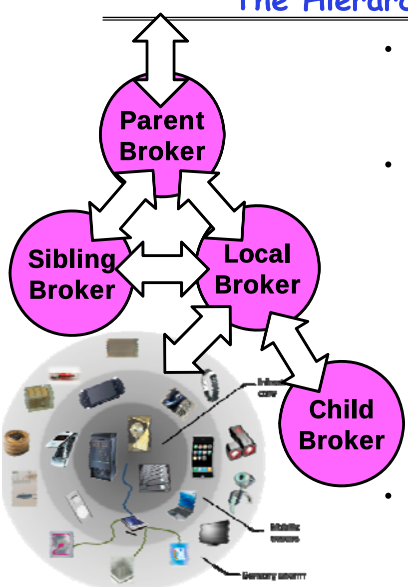


## Resource Discovery and Ontology

- Dynamically discover resources, services, and cyber-physical components (sensors/actuators) that meet application requirements
  - Find *local* components that meet some specification
  - Use ontology to describe exactly what component do
  - Distribute these resources (or fractions of services) to application cells in order to meet QoS requirements
- Many partial solutions out there, no complete solutions
  - Must deal with locality (discover local items) while at same time dealing with remote (global) services
  - Must gracefully handle failover of components
- One important aspect is that resources must be handed out only to authorized users
  - Authorization can involve ownership, micropayments, etc..



## Brokering Service: The Hierarchy of Ownership



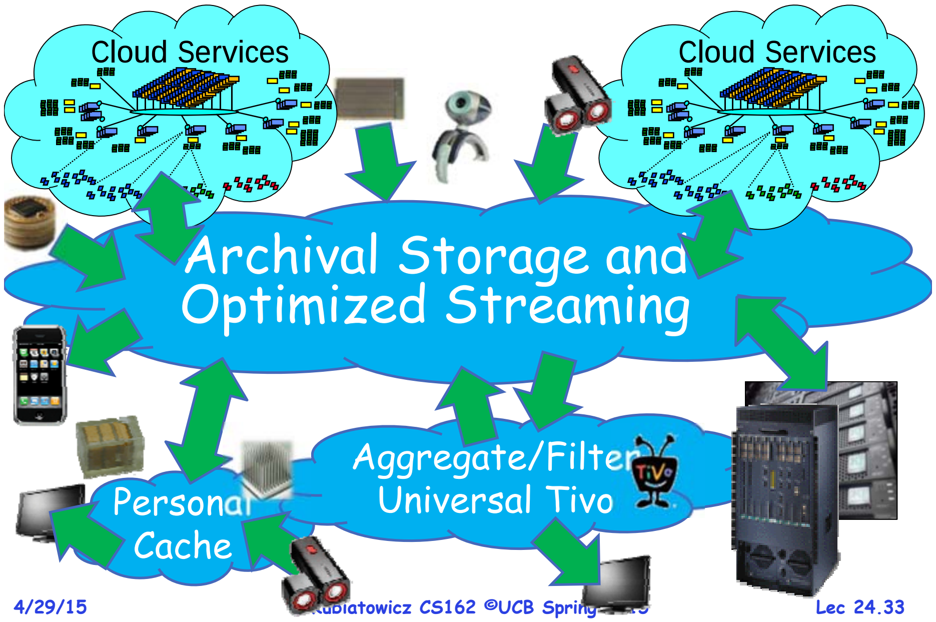

- Discover Resources in "Domain"
  - Devices, Services, Other Brokers
  - Resources self-describing?
- Allocate and Distribute Resources to Cells that need them
  - Solve Impedance-mismatch problem
  - Dynamically optimize execution
  - Hand out Service-Level Agreements (SLAs) to Cells
  - Deny admission to Cells which violate existing agreements
- Complete hierarchy
  - World graph of applications



## DataCentric Vision

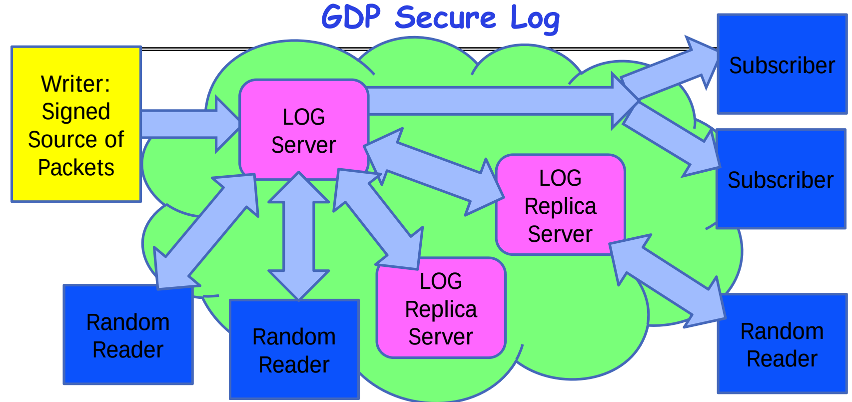
- Hardware resources are a commodity
  - Computation resource fails? Get another
  - Sensor fails? Find another
  - Change your location? Find new resources
- All that really matters is the information
  - Integrity, Privacy, Availability, Durability
  - Hardware to prevent accidental information leakage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

## The Global Data Plane





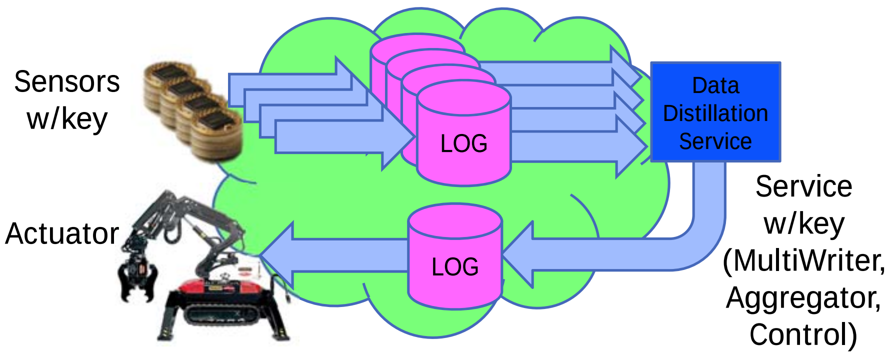

## GDP Secure Log




- Locality Optimization/QoS
  - Flat 256-bit address space
  - Routes adapted as elements move
  - Hardware QoS exploited
  - Multicast trees built as needed
- Durability
  - Replicas/Reed-Solomon coding
- Single Writer/Append only
  - Owner Key Signs entries
  - LOG server rejects bad entries
  - Tradeoff in granularity, i.e. frequency of signatures
- Multiple Readers/Subscribers
  - Random access/push based



## Simple Log-based Use-case



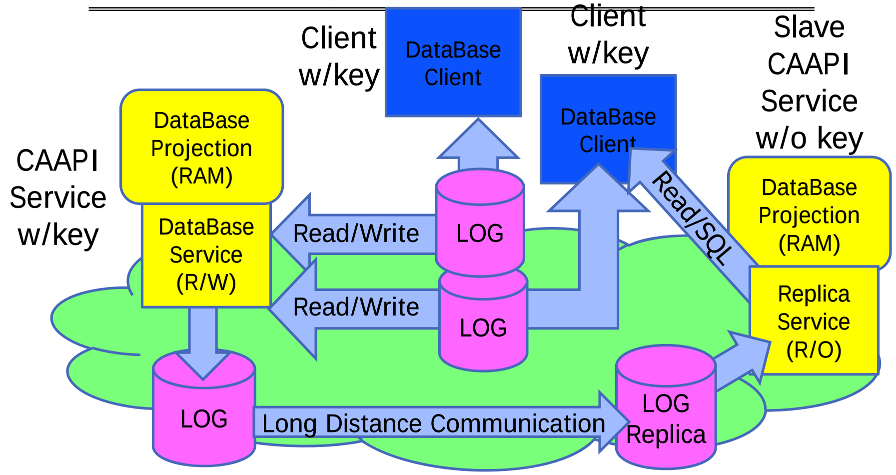

- Lightweight Logs ⇒ One Log per Device
- Log Input Secured via Owner Key/Checked by consumer
- Optional encryption for privacy
- Timestamps to help ensure freshness



## Build DataStores on top of GDP through Composition

- Common Access APIs (CAAPIs): Support common data access methods such as:
  - Key/Value Store
  - Object Store/File System
  - Data Base (i.e. Google Spanner)
- CAPPIs exported by services that consume the LOG
  - Much more convenient way to access data
- The LOG is the *Ground Truth* for data, but data is projected into a more convenient form
  - To do Random File access, Indexing, SQL queries, Latest value for given Key, etc
  - Optional Checkpoints stored for quick restart/cloning

## Example: DataBase CAAPI




- CAAPI Service can be taken down, replicated, and restarted
- Time-stamp driven transactions (Google Spanner)
- Cloud-based computation (Spark)

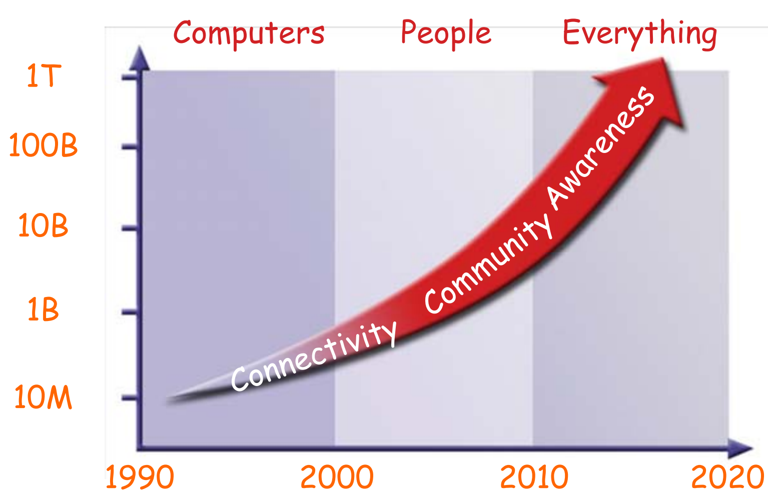


## Properties of the GDP (Summary)

- Universal way to address every stream of information
  - Publish/Subscribe view of information
  - Large *flat* address space (at least 256 bits)
  - Mechanisms for access control, privacy, and transactions
  - Streams of data persisted automatically for later access
- Location Independence ⇒ Above network level
  - Build Swarmlets once and run them anywhere
  - Migrate or replicate *running* swarmlets
  - Locality optimization/QoS handled by underlying system
- Common Access APIs (CAAPIs) provide standard Interfaces
  - Key/Value Store, Data Bases, File Systems
- Deep Archival Storage:
  - Automatic Geographically Distributed Archival Storage
- One system for sensors and big data

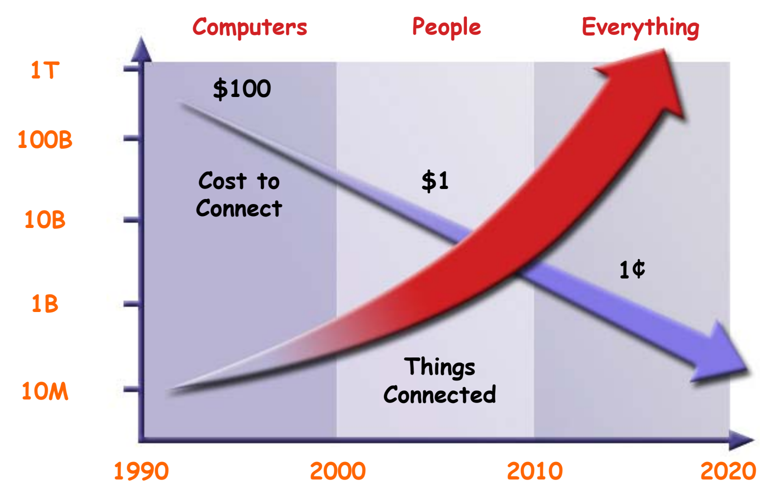


## The Revolution






## The Revolution

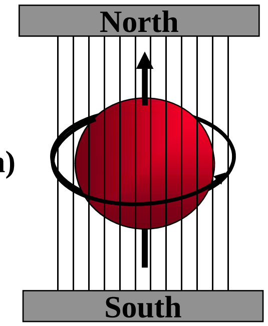


## Use Quantum Mechanics to Compute?

- Weird but useful 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.
  - Materials simulation: exponential classically, linear-time QM



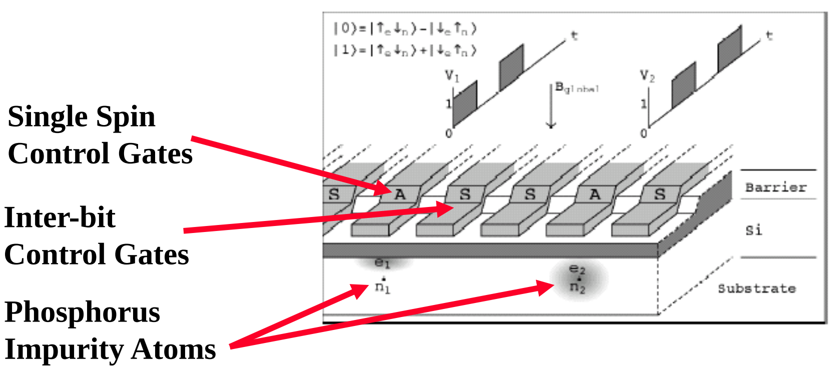
## Quantization: Use of "Spin"




- 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)




- 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 $|0\rangle$ (say "UP")
  - With probability $|C_1|^2$ we will find $|1\rangle$ (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!

## 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:

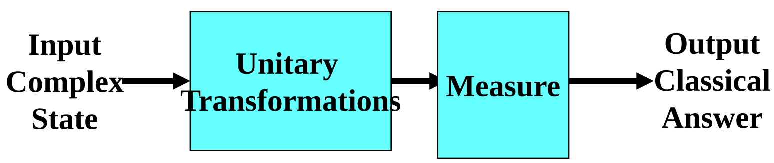
$$\Psi = C_{00}|00\rangle + C_{11}|11\rangle$$

  - 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 2n inputs equally probable!
  - Unitary transformations compute on coefficients
    - » Must maintain probability property (sum of squares = 1)
    - » Looks like doing computation on all 2n inputs simultaneously!
  - Output is one result attained by measurement
- If do this poorly, just like probabilistic computation:
  - If 2n inputs equally probable, may be 2n 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



## Shor's Factoring Algorithm

- The Security of RSA Public-key cryptosystems depends on the difficulty 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)

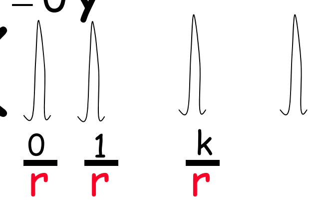
Easy 1) Choose random $x$ : $2 \le x \le N-1$.
Easy 2) If $\gcd(x,N) \ne 1$, Bingo!
Hard 3) Find smallest integer $r$ : $x^r \equiv 1 \pmod{N}$
Easy 4) If $r$ is odd, GOTO 1
Easy 5) If $r$ is even, $a \equiv x^{r/2} \pmod{N} \Rightarrow (a-1)\times(a+1) = kN$
Easy 6) If $a \equiv N-1 \pmod{N}$ GOTO 1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

## Finding r with $x^r \equiv 1 \pmod N$

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

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

$$\xrightarrow{\text{Quantum Fourier Transform}} \sum_{w=0}^{r-1}\Big( \quad \Big)|x^w\rangle$$




- 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

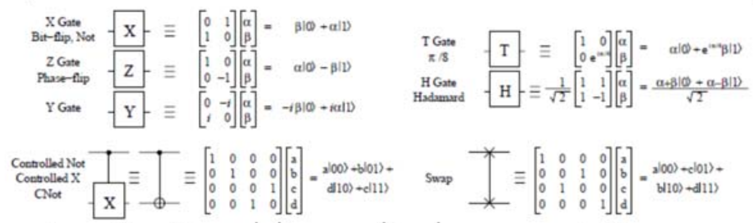


## Quantum Computing Architectures

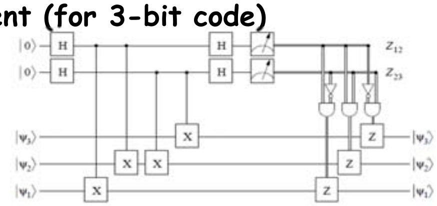
- Why study quantum computing?
  - Interesting, says something about physics
    - » Failure to build ⇒ quantum mechanics wrong?
  - Mathematical Exercise (perfectly good reason)
  - Hope that it will be practical someday:
    - » Shor's factoring, Grover's search, Design of Materials
    - » Quantum Co-processor included in your Laptop?
- To be practical, will need to hand quantum computer design off to classical designers
  - Baring Adiabatic algorithms, will probably need 100s to 1000s (millions?) of working logical Qubits ⇒ 1000s to millions of physical Qubits working together
  - Current chips: ~1 billion transistors!
- Large number of components is realm of *architecture*
  - What are optimized structures of quantum algorithms when they are mapped to a physical substrate?
  - Optimization not possible by hand
    - » Abstraction of elements to design larger circuits
    - » Lessons of last 30 years of VLSI design: USE CAD



## Quantum Circuit Model

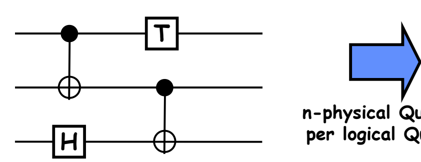



- Quantum Circuit model – graphical representation
  - Time Flows from left to right
  - Single Wires: persistent Qubits, Double Wires: classical bits
    - » Qubit – coherent combination of 0 and 1: $\psi = \alpha|0\rangle + \beta|1\rangle$
  - Universal gate set: Sufficient to form all unitary transformations
- Example: Syndrome Measurement (for 3-bit code)
  - Measurement (meter symbol) produces classical bits
- Quantum CAD
  - Circuit expressed as netlist
  - Computer manpulated circuits and implementations

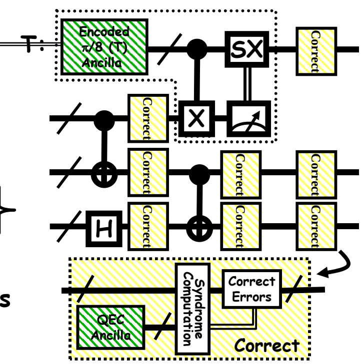




## Adding Quantum ECC



n-physical Qubits per logical Qubit




- Quantum State Fragile ⇒ encode all Qubits
  - Uses many resources: e.g. 3-level [[7,1,3]] code 343 physical Qubits/logical Qubit)!
- Still need to handle operations (fault-tolerantly)
  - *Some* set of gates are simply "transversal:"
    - » Perform identical gate between each physical bit of logical encoding
  - Others (like T gate for [[7,1,3]] code) cannot be handled transversally
    - » Can be performed fault-tolerantly by preparing appropriate ancilla
- Finally, need to perform periodical error correction
  - Correct after every(?): Gate, Long distance movement, Long Idle Period
  - Correction reducing entropy ⇒ Consumes Ancilla bits
- Observation: ≥ 90% of QEC gates are used for ancilla production
  ≥ 70-85% of all gates are used for ancilla production

## Outline

- Quantum Computing
- Ion Trap Quantum Computing
- Quantum Computer Aided Design
  - Area-Delay to Correct Result (ADCR) metric
  - Comparison of error correction codes
- Quantum Data Paths
  - QLA, CQLA, Qalypso
  - Ancilla factory and Teleportation Network Design
- Error Correction Optimization ("Recorrection")
- Shor's Factoring Circuit Layout and Design



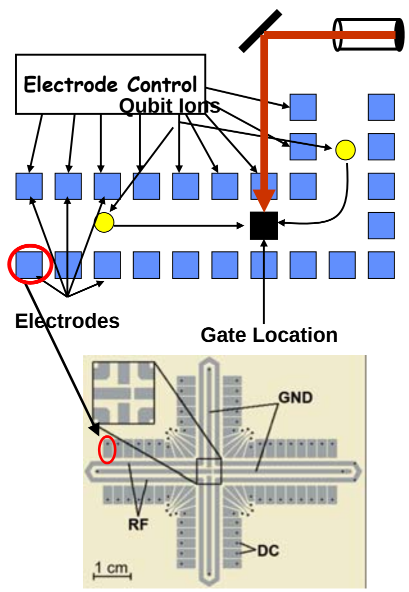
## MEMs-Based Ion Trap Devices

- Ion Traps: One of the more promising quantum computer implementation technologies
  - Built on Silicon
    - » Can bootstrap the vast infrastructure that currently exists in the microchip industry
  - Seems to be on a "Moore's Law" like scaling curve
    - » 12 bits exist, 30 promised soon, …
    - » Many researchers working on this problem
  - Some optimistic researchers speculate about room temperature
- Properties:
  - Has a long-distance Wire
    - » So-called "ballistic movement"
  - Seems to have relatively long decoherence times
  - Seems to have relatively low error rates for:
    - » Memory, Gates, Movement



## Quantum Computing with Ion Traps

- Qubits are atomic ions (e.g. $Be^{+}$)
  - State is stored in hyperfine levels
  - Ions suspended in channels between electrodes
- Quantum gates performed by lasers (either one or two bit ops)
  - Only at certain trap locations
  - Ions move between laser sites to perform gates
- Classical control
  - Gate (laser) ops
  - Movement (electrode) ops
    - Complex pulse sequences to cause Ions to migrate
    - Care must be taken to avoid disturbing state
- Demonstrations in the Lab
  - NIST, MIT, Michigan, many others

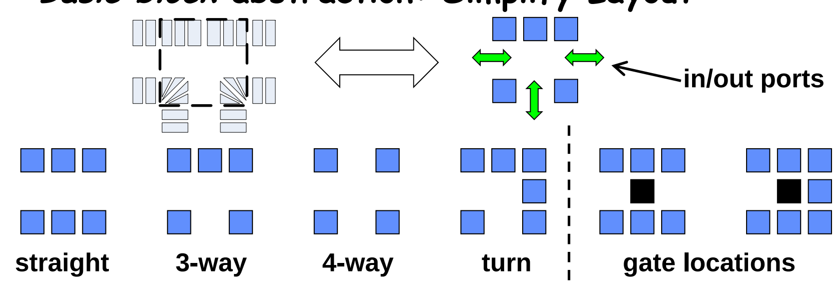



Courtesy of Chuang group, MIT



## An Abstraction of Ion Traps

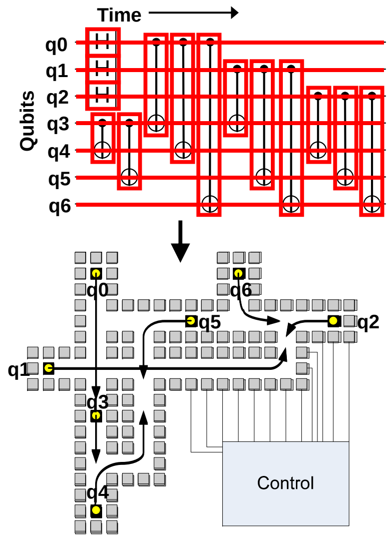
- *Basic block* abstraction: Simplify Layout




- Evaluation of layout through simulation
  - Yields Computation Time and Probability of Success
- Simple Error Model: Depolarizing Errors
  - Errors for every Gate Operation and Unit of Waiting
  - Ballistic Movement Error: Two error Models
    1. Every Hop/Turn has probability of error
    2. Only Accelerations cause error

## Ion Trap Physical Layout

- Input: Gate level quantum circuit
  - Bit lines
  - 1-qubit gates
  - 2-qubit gates
- Output:
  - Layout of channels
  - Gate locations
  - Initial locations of ions
  - Movement/gate schedule
  - Control for schedule



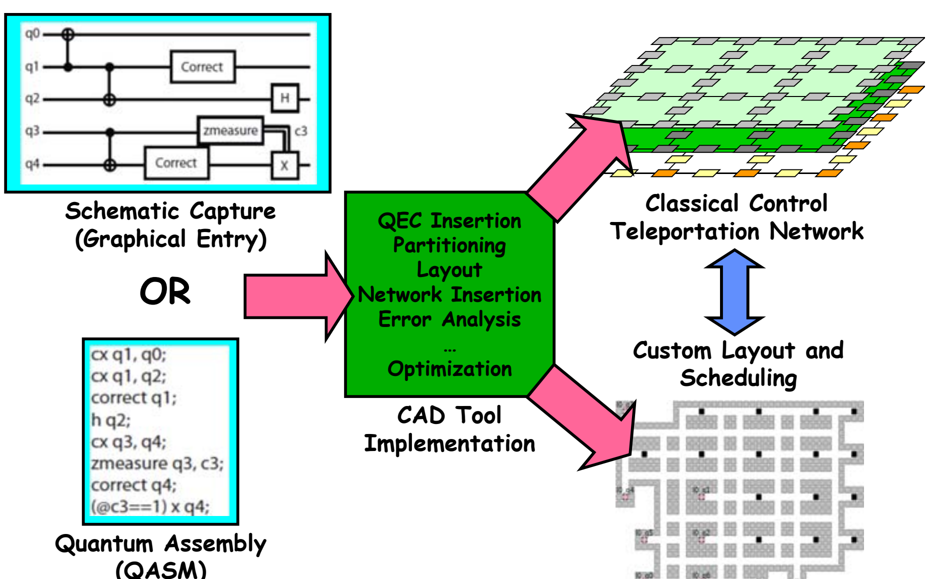



## Outline

- Quantum Computering
- Ion Trap Quantum Computing
- Quantum Computer Aided Design
  - Area-Delay to Correct Result (ADCR) metric
  - Comparison of error correction codes
- Quantum Data Paths
  - QLA, CQLA, Qalypso
  - Ancilla factory and Teleportation Network Design
- Error Correction Optimization ("Recorrection")
- Shor's Factoring Circuit Layout and Design



## Vision of Quantum Circuit Design





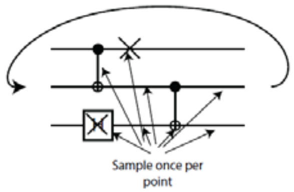

## Important Measurement Metrics

- Traditional CAD Metrics:
  - Area
    » What is the total area of a circuit?
    » Measured in macroblocks (ultimately $\mu m^2$ or similar)
  - Latency ($Latency_{single}$)
    » What is the total latency to compute circuit *once*
    » Measured in seconds (or $\mu s$)
  - 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
  - $ADCR = Area \times E(Latency) = \dfrac{Area \times Latency_{single}}{P_{success}}$
  - $ADCR_{optimal}$: Best ADCR over all configurations
- Optimization potential: Equipotential designs
  - Trade Area for lower latency
  - Trade lower probability of success for lower latency

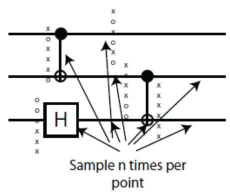
## How to evaluate a circuit?

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

Normal Monte Carlo: n times



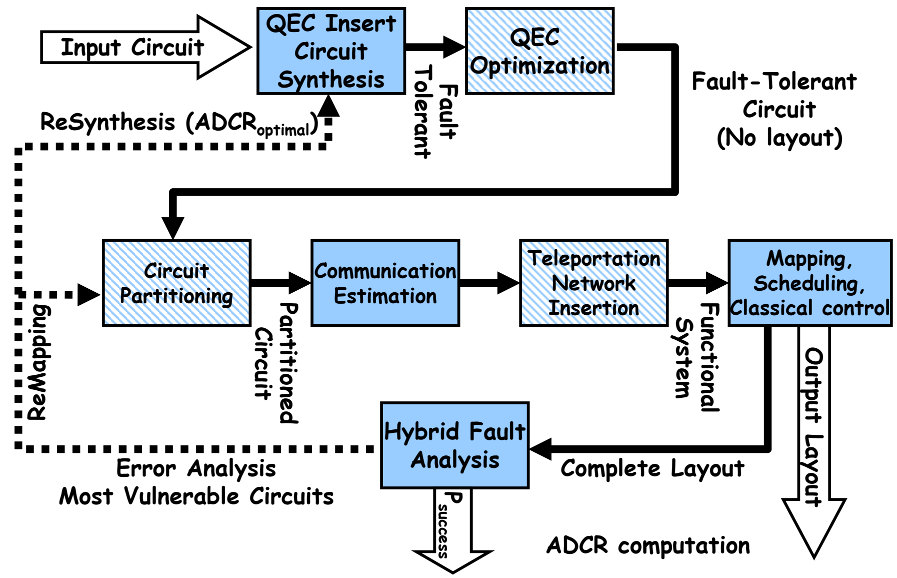

Vector Monte Carlo: single pass




- 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
    - » Smaller modules evaluated via vector Monte Carlo
    - » Teleportation infrastructure evaluated via fidelity of EPR bits
- Finally – Compute ADCR for particular result



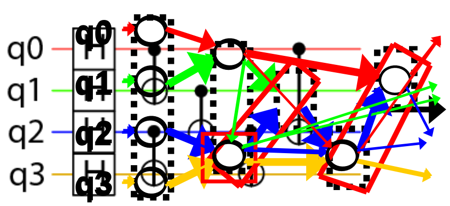
## Quantum CAD flow

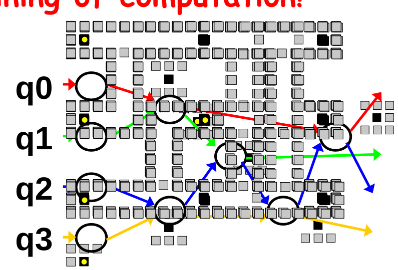





## Example Place and Route Heuristic: Collapsed Dataflow

- Gate locations placed in dataflow order
  - Qubits flow left to right
  - Initial dataflow geometry folded and sorted
  - Channels routed to reflect dataflow edges
- Too many gate locations, collapse dataflow
  - Using scheduler feedback, identify latency critical edges
  - Merge critical node pairs
  - Reroute channels
- Dataflow mapping allows pipelining of computation!



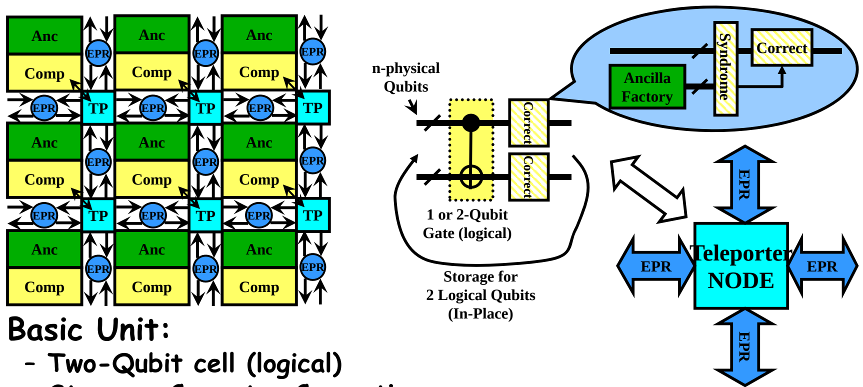






## Outline

- Quantum Computing
- Ion Trap Quantum Computing
- Quantum Computer Aided Design
  - Area-Delay to Correct Result (ADCR) metric
  - Comparison of error correction codes
- Quantum Data Paths
  - QLA, CQLA, Qalypso
  - Ancilla factory and Teleportation Network Design
- Error Correction Optimization ("Recorrection")
- Shor's Factoring Circuit Layout and Design

## Quantum Logic Array (QLA)

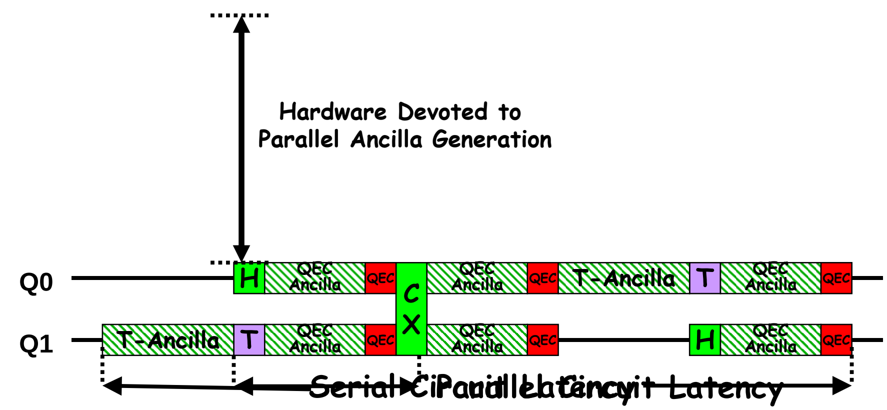



- Basic Unit:
  - Two-Qubit cell (logical)
  - Storage, Compute, Correction
- Connect Units with Teleporters
  - Probably in mesh topology, but details never entirely clear from original papers
- First Serious (Large-scale) Organization (2005)
  - Tzvetan S. Metodi, Darshan Thaker, Andrew W. Cross, Frederic T. Chong, and Isaac L. Chuang



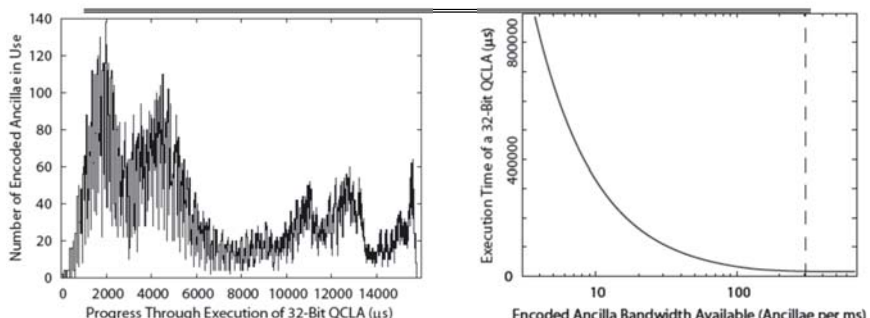
## Running Circuit at "Speed of Data"

- Often, Ancilla qubits are independent of data
  - Preparation may be pulled offline
  - Very clear Area/Delay tradeoff:
- Ancilla qubits should be ready "just in time" to avoid ancilla decoherence from idleness






## How much Ancilla Bandwidth Needed?

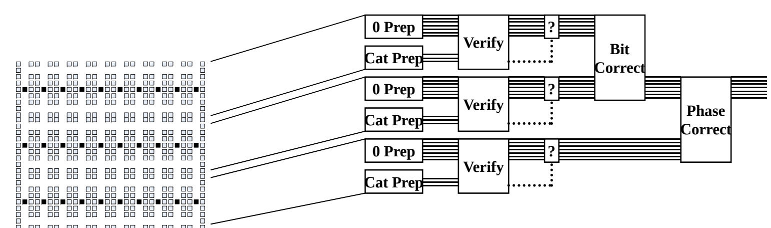



- 32-bit Quantum Carry-Lookahead Adder
  - Ancilla use very uneven (zero and T ancilla)
  - Performance is flat at high end of ancilla generation bandwidth
    - » Can back off 10% in maximum performance an save orders of magnitude in ancilla generation area
- Many bits idle at any one time
  - Need only enough ancilla to maintain state for these bits
  - Many not need to frequently correct idle errors
- Conclusion: makes sense to compute ancilla requirements and share area devoted to ancilla generation

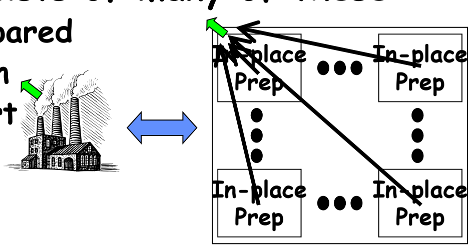


## Ancilla Factory Design I

- "In-place" ancilla preparation

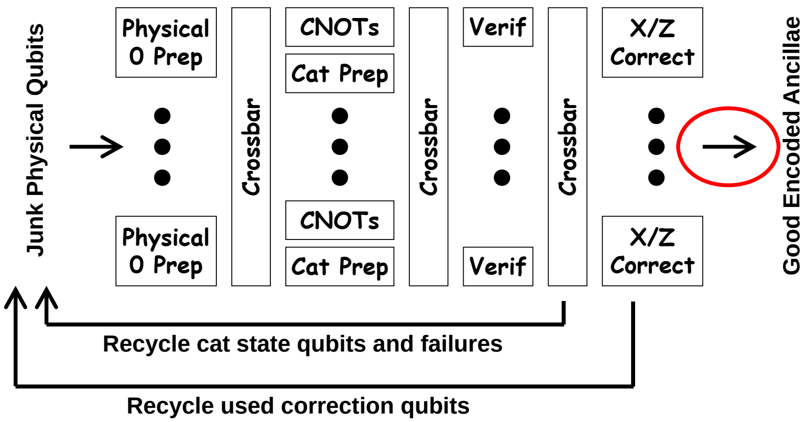



- Ancilla factory consists of many of these
  - Encoded ancilla prepared in many places, then moved to output port
  - Movement is costly!



## Ancilla Factory Design II

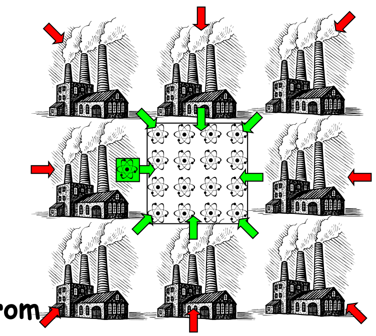
- Pipelined ancilla preparation: break into stages
  - Steady stream of encoded ancillae at *output port*
  - Fully laid out and scheduled to get area and bandwidth estimates



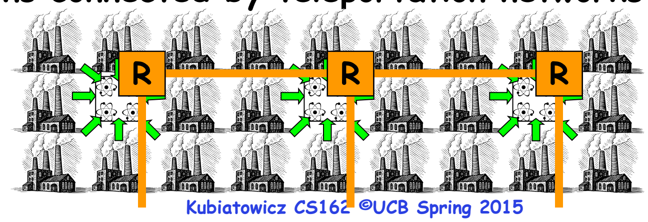



## The Qalypso Datapath Architecture

- Dense data region
  - Data qubits *only*
  - Local communication
- Shared Ancilla Factories
  - Distributed to data as needed
  - Fully multiplexed to all data
  - Output ports (⇨): close to data
  - Input ports (➡): may be far from data (recycled state irrelevant)

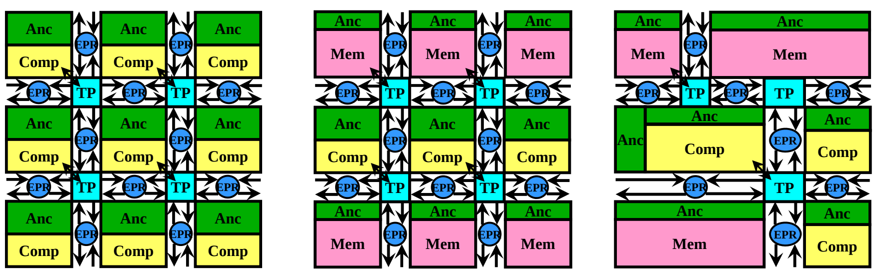


- Regions connected by teleportation networks






## Tiled Quantum Datapaths



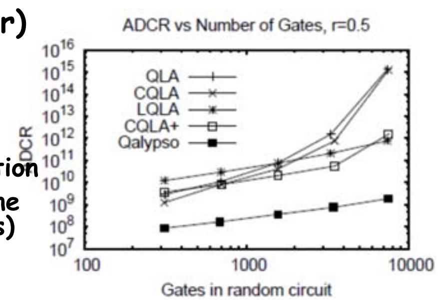

Previous: QLA, LQLA    Previous: CQLA, CQLA+    Our Group: Qalypso

- Several Different Datapaths mappable by our CAD flow
  - Variations include hand-tuned Ancilla generators/factories
- Memory: storage for state that doesn't move much
  - Less/different requirements for Ancilla
  - Original CQLA paper used different QEC encoding
- Automatic mapping must:
  - Partition circuit among compute and memory regions
  - Allocate Ancilla resources to match demand (at knee of curve)
  - Configure and insert teleportation network



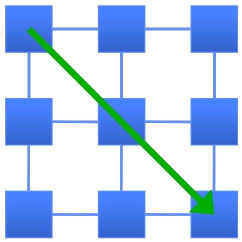
## Which Datapath is Best?

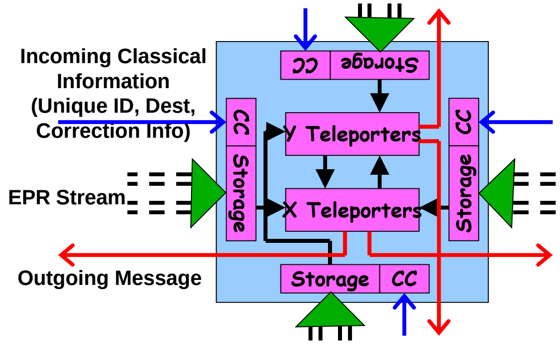
- Random Circuit Generation
  - f(Gate Count, Gate Types, Qubit Count, Splitting factor)
  - Splitting factor (r): measures connectivity of the circuit
    - » Example: 0.5 splits Qubits in half, adds random gates between two halves, then recursively splits results
    - » Closely related to Rent's parameter
- Qalypso clear winner (for all r)
  - 4x lower latency than LQLA
  - 2x smaller area than CQLA+
- Why Qalypso does well:
  - Shared, matched ancilla generation
  - Automatic network sizing (*not* one Teleporter for every two Qubits)
  - Automatic Identification of Idle Qubits (memory)




- LQLA and CQLA+ perform close second
  - Original datapaths supplemented with better ancilla generators, automatic network sizing, and Idle Qubit identification
  - Original QLA and CQLA do very poorly for large circuits

## How to design Teleportation Network

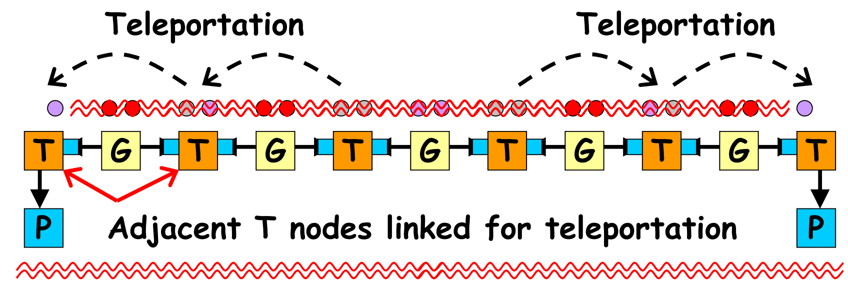





- What is the architecture of the network?
  - Including Topology, Router design, EPR Generators, etc..
- What are the details of EPR distribution?
- What are the practical aspects of routing?
  - When do we set up a channel?
  - What path does the channel take?



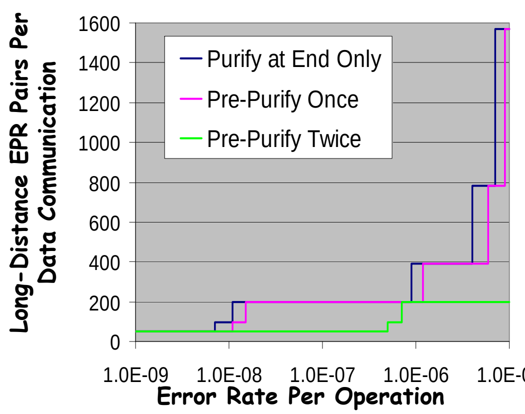
## Basic Idea: Chained Teleportation

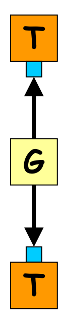



- Positive Features
  - Regularity (can build classical network topologies)
  - T node linking not on critical path
  - Pre-purification part of link setup
    - » Fidelity amplification of the line
  - Allows continuous stream of EPR correlations to be established for use when necessary



## Pre-Purification

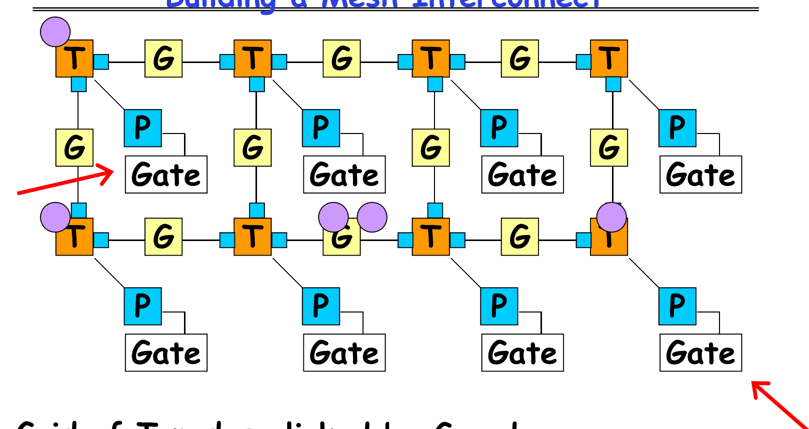



- Experiment: Transmit enough EPR pairs over network to meet required fidelity of channel
  - Measure total global traffic
  - Higher Fidelity local EPR pairs ⇒ less global EPR traffic
- Benefit: decreased congestion at T Nodes



## Building a Mesh Interconnect



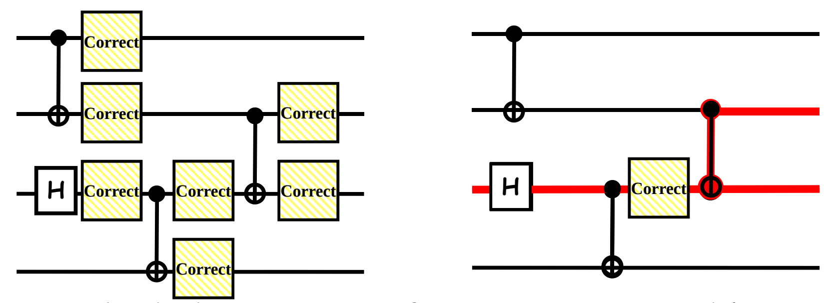

- Grid of T nodes, linked by G nodes
- Packet-switched network
  - Options: Dimension-Order or Adaptive Routing
  - Precomputed or on-demand start time for setup
- Each EPR qubit has associated classical message

## Outline

- Quantum Computing
- Ion Trap Quantum Computing
- Quantum Computer Aided Design
  - Area-Delay to Correct Result (ADCR) metric
  - Comparison of error correction codes
- Quantum Data Paths
  - QLA, CQLA, Qalypso
  - Ancilla factory and Teleportation Network Design
- Error Correction Optimization ("Recorrection")
- Shor's Factoring Circuit Layout and Design



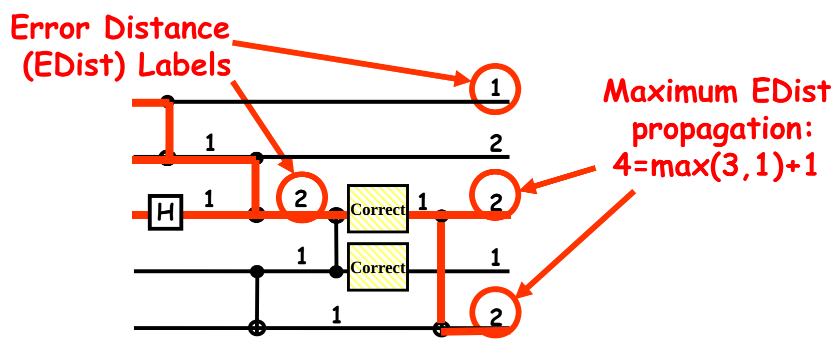
## Reducing QEC Overhead




- Standard idea: correct after every gate, and long communication, and long idle time
  - This is the easiest for people to analyze
- This technique is suboptimal (at least in some domains)
  - Not every bit has same noise level!
- Different idea: identify critical Qubits
  - Try to identify paths that feed into noisiest output bits
  - Place correction along these paths to reduce maximum noise



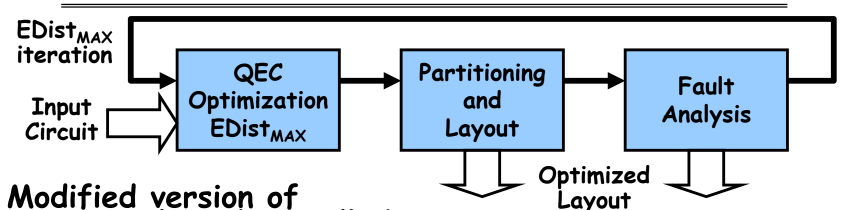
## Simple Error Propagation Model




- EDist model of error propagation:
  - Inputs start with EDist = 0
  - Each gate propagates max input EDist to outputs
  - Gates add 1 unit of EDist, Correction resets EDist to 1
- Maximum EDist corresponds to Critical Path
  - Back track critical paths that add to Maximum EDist
- Add correction to keep EDist below critical threshold

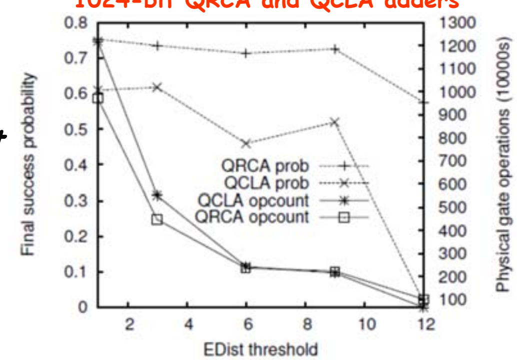


## QEC Optimization

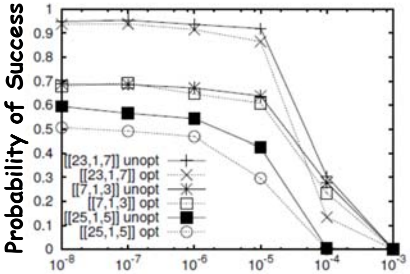



- Modified version of *retiming* algorithm: called "recorrection:"
  - Find minimal placement of correction operations that meets specified MAX(EDist) ≤ $EDist_{MAX}$
- Probably of success *not* always reduced for $EDist_{MAX} > 1$
  - But, operation count and area drastically reduced
- Use Actual Layouts and Fault Analysis
  - Optimization *pre-layout*, evaluated *post-layout*

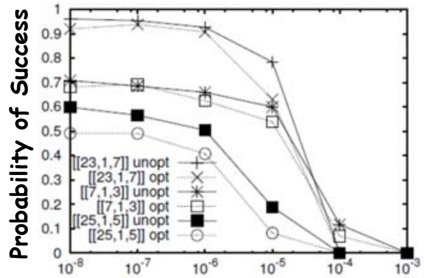
1024-bit QRCA and QCLA adders



## Recorrection in presence of different QEC codes




Move Error Rate per Macroblock
$EDist_{MAX}=3$




Idle Error Rate per CNOT Time
$EDist_{MAX}=3$

- 500 Gate Random Circuit (r=0.5)
- Not all codes do equally well with Recorrection
  - Both [[23,1,7]] and [[7,1,3]] reasonable candidates
  - [[25,1,5]] doesn't seem to do as well
- Cost of communication and Idle errors is clear here!
- However – real optimization situation would vary EDist to find optimal point



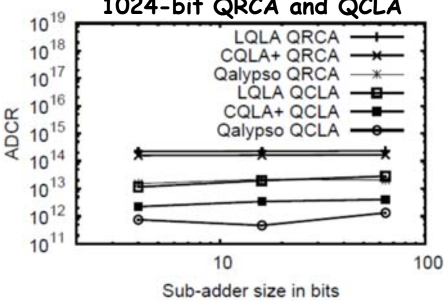
## Outline

- Quantum Computing
- Ion Trap Quantum Computing
- Quantum Computer Aided Design
  - Area-Delay to Correct Result (ADCR) metric
  - Comparison of error correction codes
- Quantum Data Paths
  - QLA, CQLA, Qalypso
  - Ancilla factory and Teleportation Network Design
- Error Correction Optimization ("Recorrection")
- Shor's Factoring Circuit Layout and Design

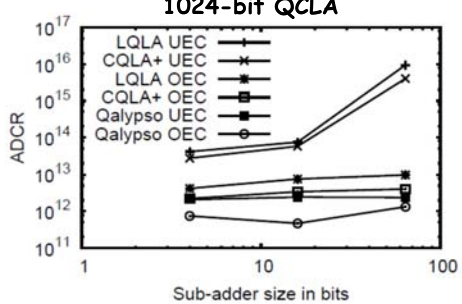


## Comparison of 1024-bit adders

$ADCR_{optimal}$ for 1024-bit QRCA and QCLA



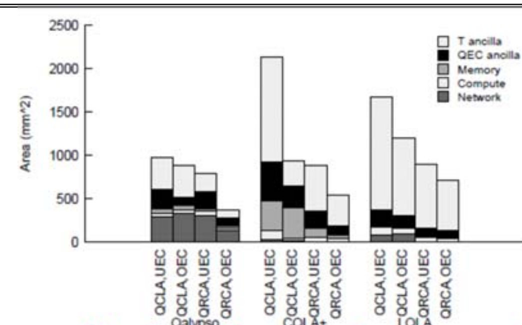

$ADCR_{optimal}$ for 1024-bit QCLA




- 1024-bit Quantum Adder Architectures
  - Ripple-Carry (QRCA)
  - Carry-Lookahead (QCLA)
- Carry-Lookahead is better in all architectures
- QEC Optimization improves ADCR by order of magnitude in some circuit configurations

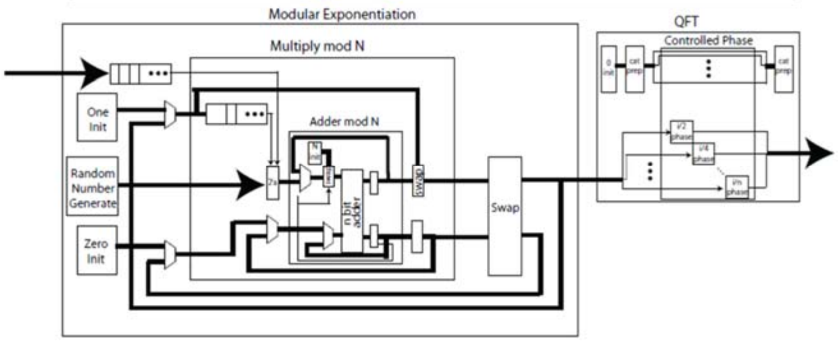


## Area Breakdown for Adders




- 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

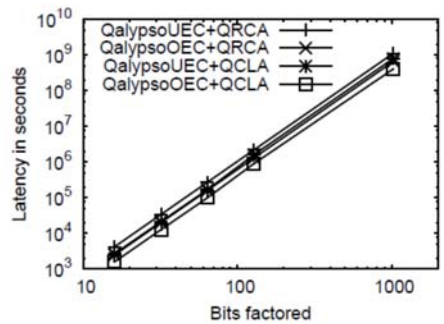
## Investigating 1024-bit Shor's

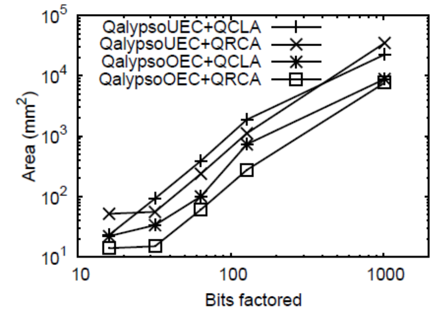



- Full Layout of all Elements
  - Use of 1024-bit Quantum Adders
  - Optimized error correction
  - Ancilla optimization and Custom Network Layout
- Statistics:
  - Unoptimized version: $1.35\times10^{15}$ operations
  - Optimized Version 1000X smaller
  - QFT is only 1% of total execution time



## 1024-bit Shor's Continued




- Circuits too big to compute $P_{success}$
  - Working on this problem
- Fastest Circuit: $6\times10^8$ seconds ~ 19 years
  - Speedup by classically computing recursive squares?
- Smallest Circuit: 7659 $mm^2$
  - Compare to previous *estimate* of 0.9 $m^2$ = $9\times10^5$ $mm^2$