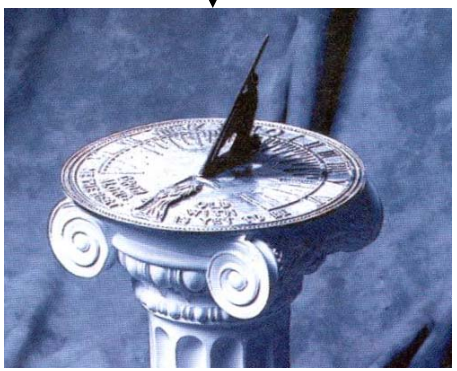
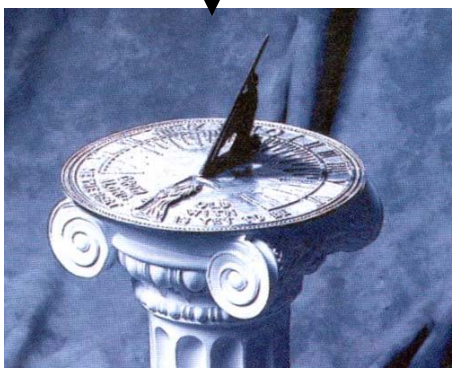



18
Distributed Time

Distributed Embedded Systems

Philip Koopman

Nov. 9, 2015

Carnegie Mellon

Preview

- **Distributed time**
  - When things happen; chain of causal events
  - Relating messages to causality

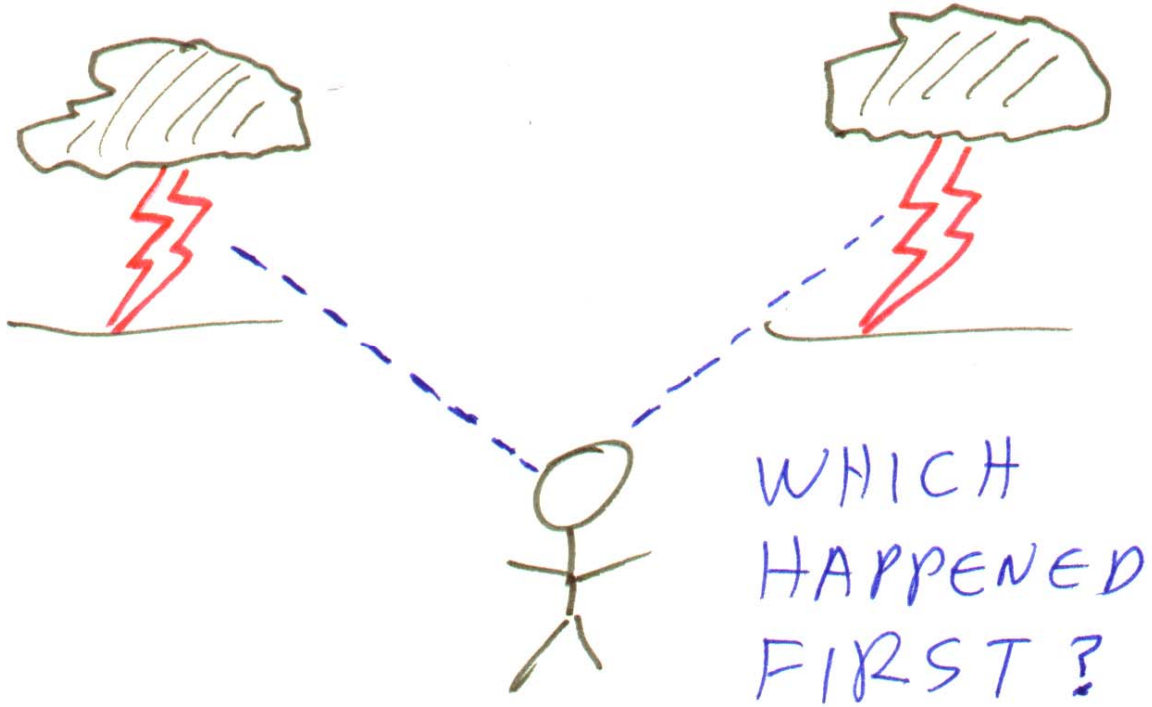
- **Clocks and time ticks**
  - It is fundamentally impossible for all nodes to have *exactly* the same clock time
  - Limitations of network messaging affect clock synchronization
    (if you use the same network to distribute time and announce events, it is difficult to have a time base more precise than event announcement jitter)

- **Clock synchronization algorithms**
  - Heavy duty distributed time algorithms are grad.-level material

- **Kopetz book: Chapter 3 goes into graduate-level descriptions**
  - The ideas are (almost) all in this lecture, but with more intuitive explanations

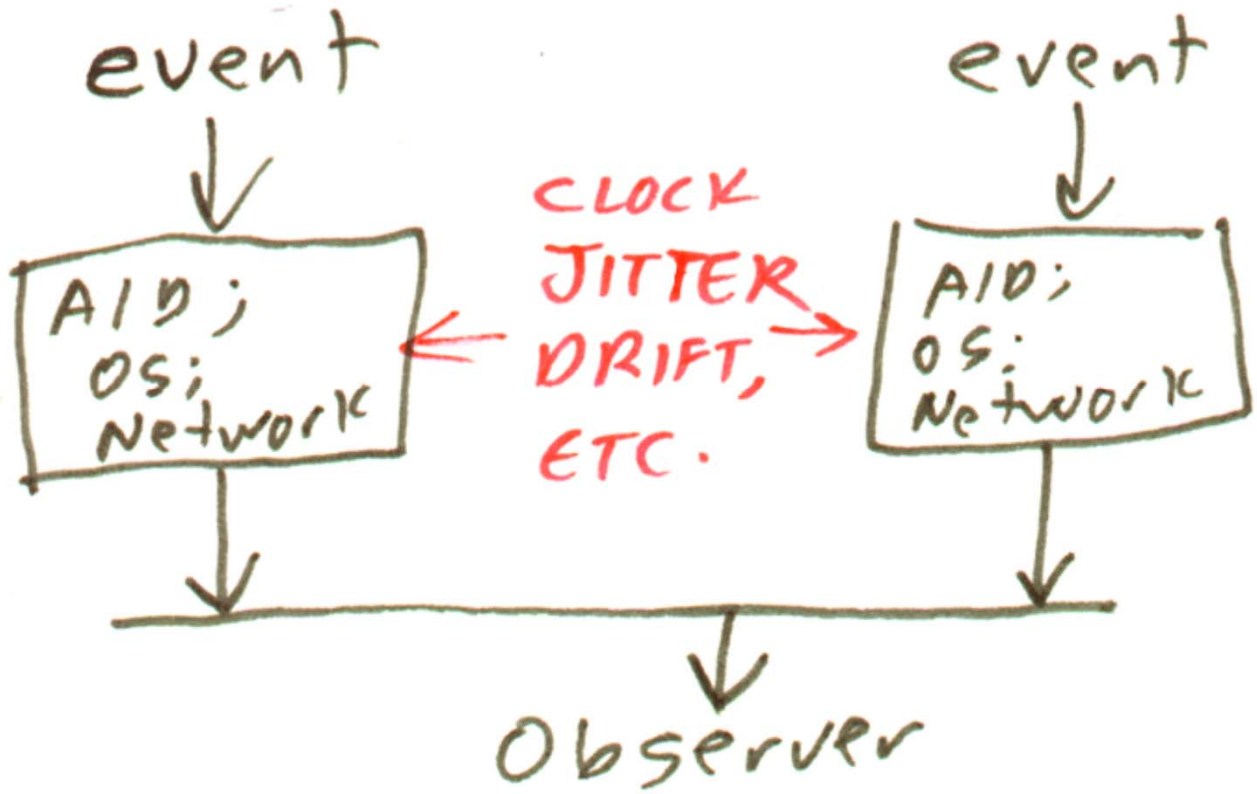
Why Is Distributed Time Difficult?

- **Fundamentally, it is impossible to have a perfect, common time base**
  - So, we hope relativistic effects don't matter
  - We put in hacks for network delay time
    - Measure typical propagation delays
    - Measure typical time variations (drift, jitter)
    - Assume that they don't change a lot, and add in fudge factors to account for them
  - But, ***EVEN THEN***, "closely spaced" events are always a problem



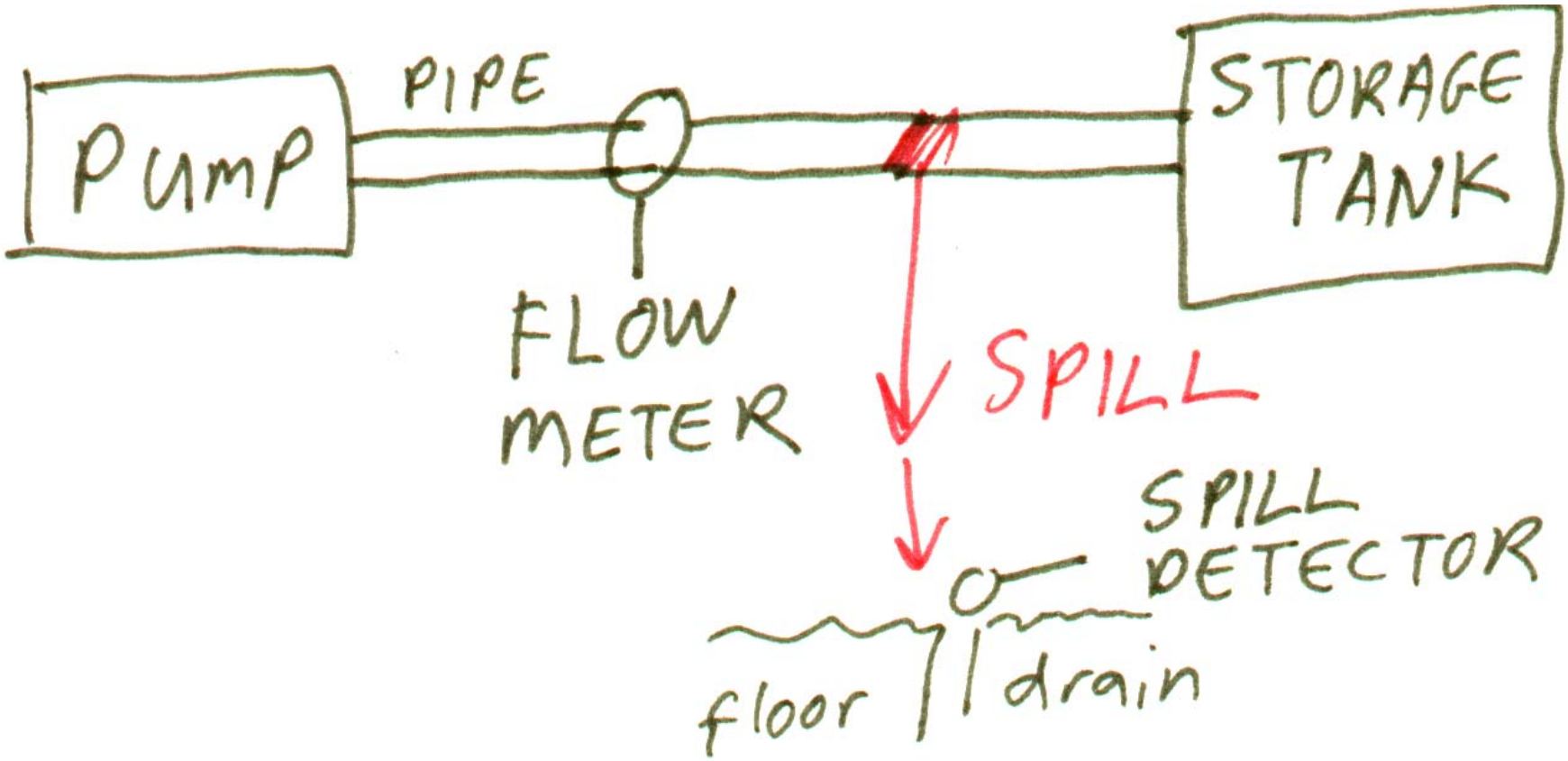
The Same Problem on a Network

- **Variations in time between event and sending network package**
  - A/D conversion speeds
  - Sampling jitter (interrupt priority/interference OR polling loop time delay)
  - Operating system task scheduling jitter
  - Network interface jitter & message prioritization



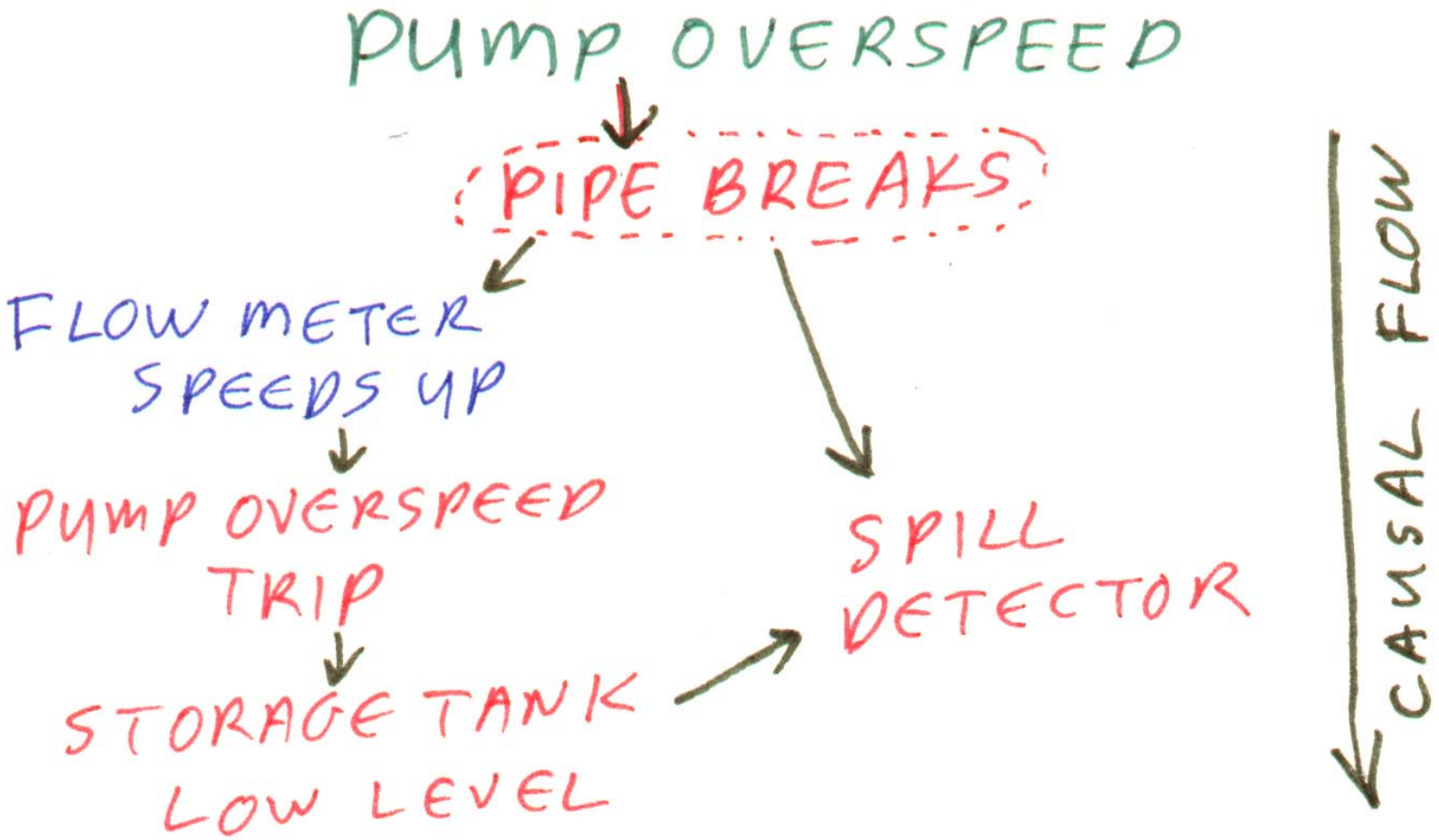
Causality & Time -- An Example

- **Let's say that a pipe breaks, causing a spill**



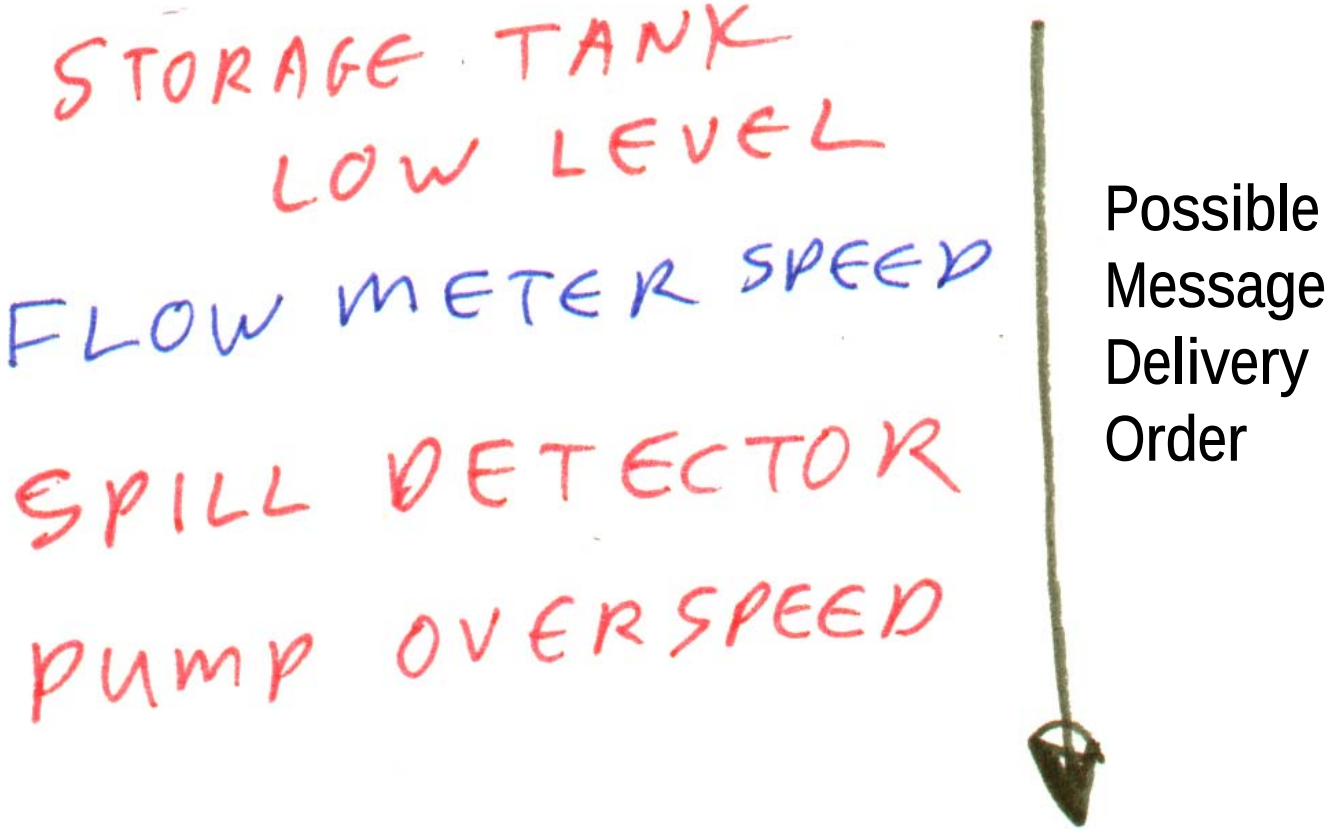
Causal Order -- How Did It Happen?

- **Pump overspeed caused pressure and vibration, breaking the pipe**
  - Assume the storage tank was full, so input flow zero despite pump
  - (Assume the pipe was defective, and overspeed stressed the latent defect)
- **Once the pipe is broken and pump stops, the tank empties**



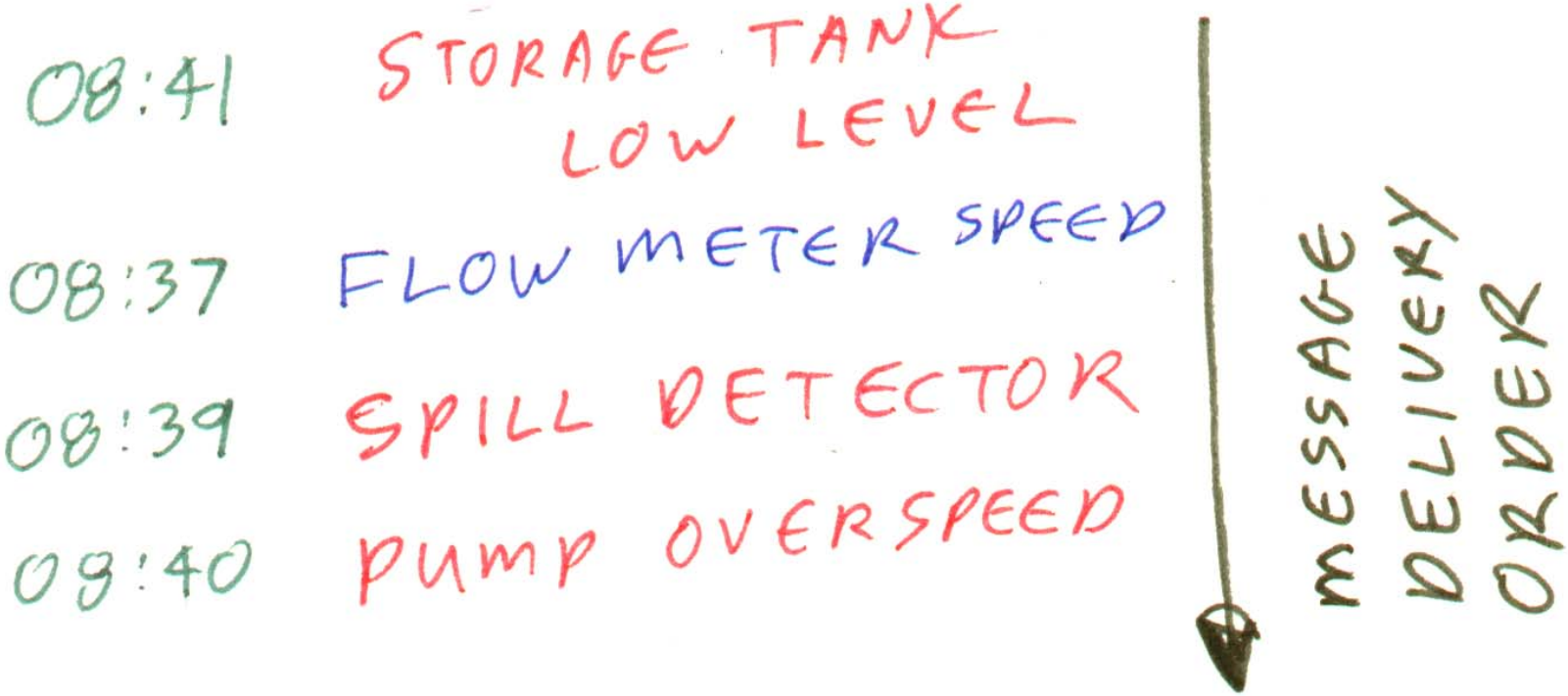
Message Delivery Order -- Can Be Arbitrary

- **Messages can be delayed by:**
  - Priority-based blocking (*e.g.*, if storage tank low level has highest priority)
  - Multi-hop routing (*e.g.*, if pump overspeed must traverse 5 network bridges)
  - Other system loads (*e.g.*, if pump overspeed computer is doing a non-interruptible computation)



An Obvious Solution -- Time Stamps

- **This is why time matters on a distributed system**
  - Provides a global sense of when things happened
  - Provides notion of dead time from sense to actuate for distributed control loops



- **BUT:**
  - Consumes bandwidth
  - Requires *synchronized* time-of-day at every node

A Less Obvious Solution – Implicit Time Stamps

- **Send every message in a cyclic pattern (Another Time Triggered Idea)**
  - Cycle #1: Message #1
  - Cycle #1: Message #2
  - Cycle #1: Message #3
  - **Cycle #2**: Message #1
  - …

- **Ensure that each message has the most recent data value when it is sent**
  - Best case is that message has brand new data
  - Worst case is that message has data almost one cycle old
  - Implicit time stamps consist of fact that data is never more than 1 cycle out of date (and, possibly, have an explicit cycle number attached)
    - This is the basis for the S 2G precedence discussion in Kopetz
    - The kinder, gentler version follows

  - But first, a word about accuracy, precision & time ticks

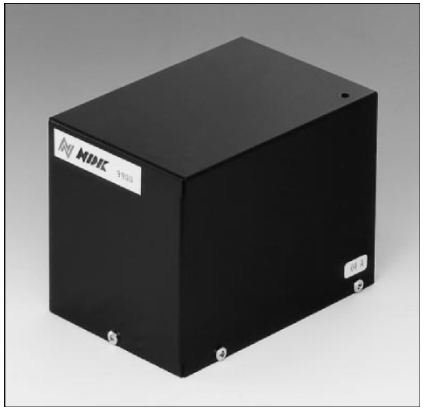
Time Measurement Inaccuracy Sources

- **Variations**
  - Synchronization difference (impossible to sync all clocks *exactly*)
  - Clock drift (too fast, too slow, maybe time-varying)

- **Quantization effects**
  - Micro-tick size limitation on a single node
  - Across-network Tick size limitation on a system

Physical Clock

- **Typical source: oscillator circuit, perhaps augmented with GPS time signal**
  - R/C timing circuit; somewhat stable
  - Commodity crystal oscillator; perhaps 10^{-6} /sec stability (14-pin DIP size)
    - Oven-controlled for wireless communications; perhaps 10^{-11} /sec stability
  - Micro-rubidium atomic oscillator
    - perhaps 10^{-11} /*month* stability
    - 0.7 kg weight
    - 0.3 liter volume



Simple Real-World Clock Drift Example

- **A gizmo has a crystal oscillator running at 32,768 Hz + 0.002%**
  - 32,768 Hz is a quartz watch crystal; 15-bit divider gives 1 Hz
  - (.002% is a $2*10^{-5}$ drift rate)
  - The product specification requires accuracy of 2 seconds/day
  - Will the oscillator meet the specification?

(.00002 sec/sec drift rate * (60 sec * 60 min * 24 hr)

= 1.728 sec drift per day (*so it meets the spec.*)

  - How far will it drift over a 2-year battery life?

1.728 sec/day * (365.25 days * 2 years) = 21 minutes drift over 2 years

- **Observations:**
  - 10^{-6} or 10^{-7} is probably desirable for consumer products that keep time
  - There are a lot of seconds in a year (31.6 million of them)
    - Roughly Pi * 10^{7}

Distributed Time Ticks

- ***Micro-tick:* granularity of clock (counter/timer interrupts)**
  - Length of time used on single node for time keeping
  - Multiple of the local process oscillator speed

- ***Global Tick:* clock events generated in global time**
  - Length of time used to coordinate events across nodes
  - Usually many micro-ticks per global tick
  - Sometimes called a *Macro-tick*
  - A network-wide notion of time, independent of micro-tick size

- **Key attribute: stability over operating time**
  - Global ticks have to happen often enough to keep things from skewing too far apart at nodes

Global Time Tick

- **Global time granularity *g* is the size of global time stamp increments**
  - To be **reasonable**, *g* has to be > Precision $\mathcal{S}$
  - Events must be 2 or more ticks apart to establish unambiguous temporal order
    - Event A might be up to 1 tick faster than a notional "reference clock"
    - Event B might be up to 1 tick slower than a notional "reference clock"
    - But, difference is < 2 ticks (*i.e.*, temporal ordering certain at >= 2g time difference)
  - (For time-triggered network messages, "g" in terms of event ordering is based on message periods)

- **How long does an event last?**
  - Up to *2g* error on start of event
  - Up to *2g* error at end of event
  - (Assumes that event start, event end, and elapsed time observer happen at different nodes)

Example, When Does the Bus Come?

(A gentle version of *S2G precedence*)

- **Say that the 61C bus comes every 5 minutes per the schedule**
  - 8:05 bus
  - 8:10 bus
  - 8:15 bus

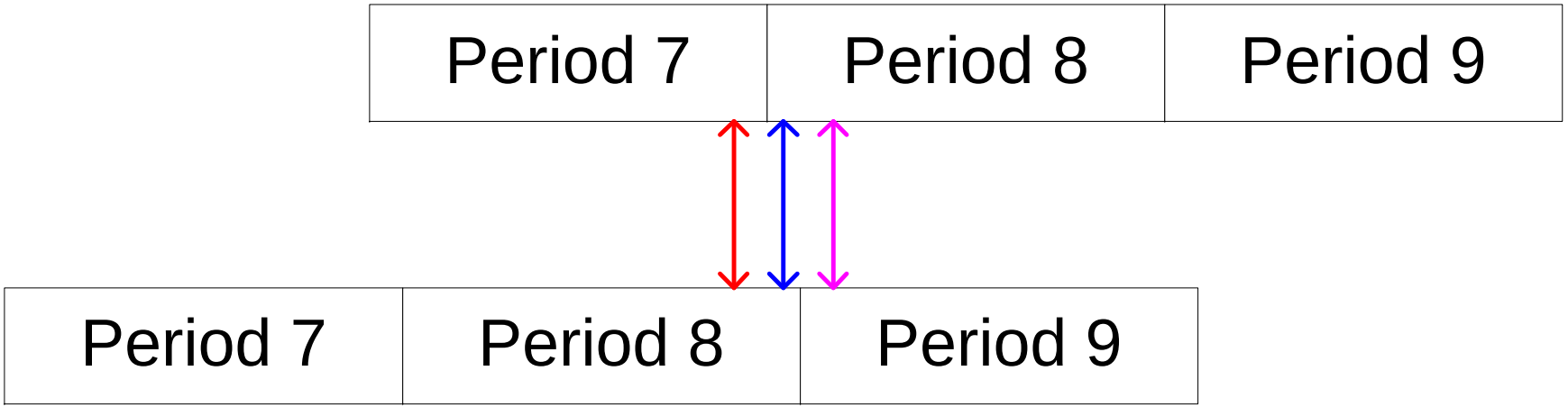
- **You get on a bus at 8:11; which bus was it?**
  - If buses run +/- 6 minutes early/late, it could have been any of the three
  - If buses run +/- 5 early/late, it was the 8:10 or the 8:15
  - If buses run < +/- 4 early/late, it could only be the 8:10

(*Moral of the story*:
don't bother with a bus schedule if average jitter exceeds scheduled inter-arrival time)

- **Now consider messages sent in a network...**
  - How do you connect message order to event sequence?

A Graphical Explanation

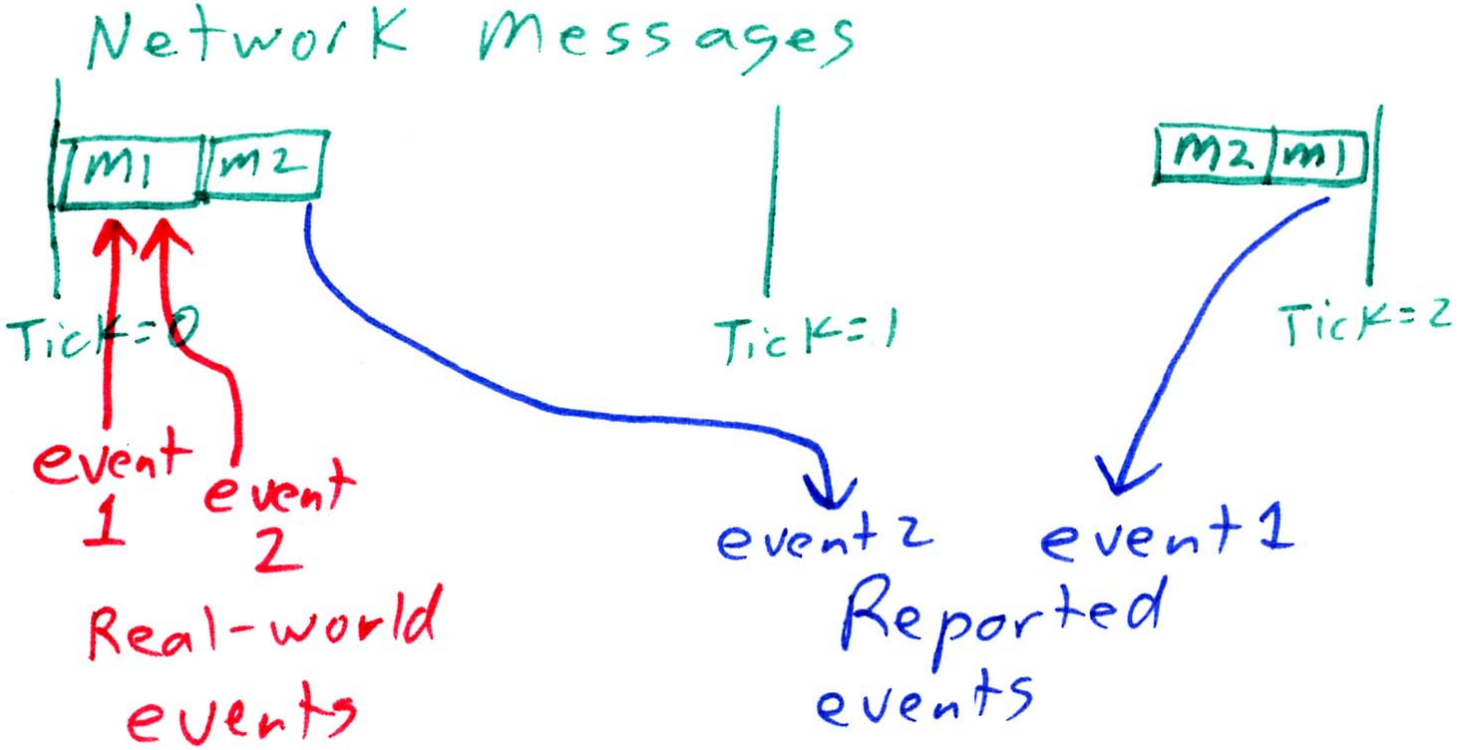
- **Assume that time period g is slightly larger than precision**
  - So there is always an instant at which all nodes have the same period number
- **An event could happen at one instant and appear to be:**
  - Period 7 in one node; period 8 in a second (leftmost vertical arrow)
  - Period 8 in both nodes
  - Period 8 in one node; period 9 in a second (rightmost vertical arrow)
  - BUT NOT: Period 7 in one, and Period 9 in a second one
  - There is no instant in which period number differ by 2 or more
    - So, any events with period numbers >= 2 apart have unambiguous order



Assume Period=g is slightly larger than precision.

Why Is It 2g Instead of 1g?

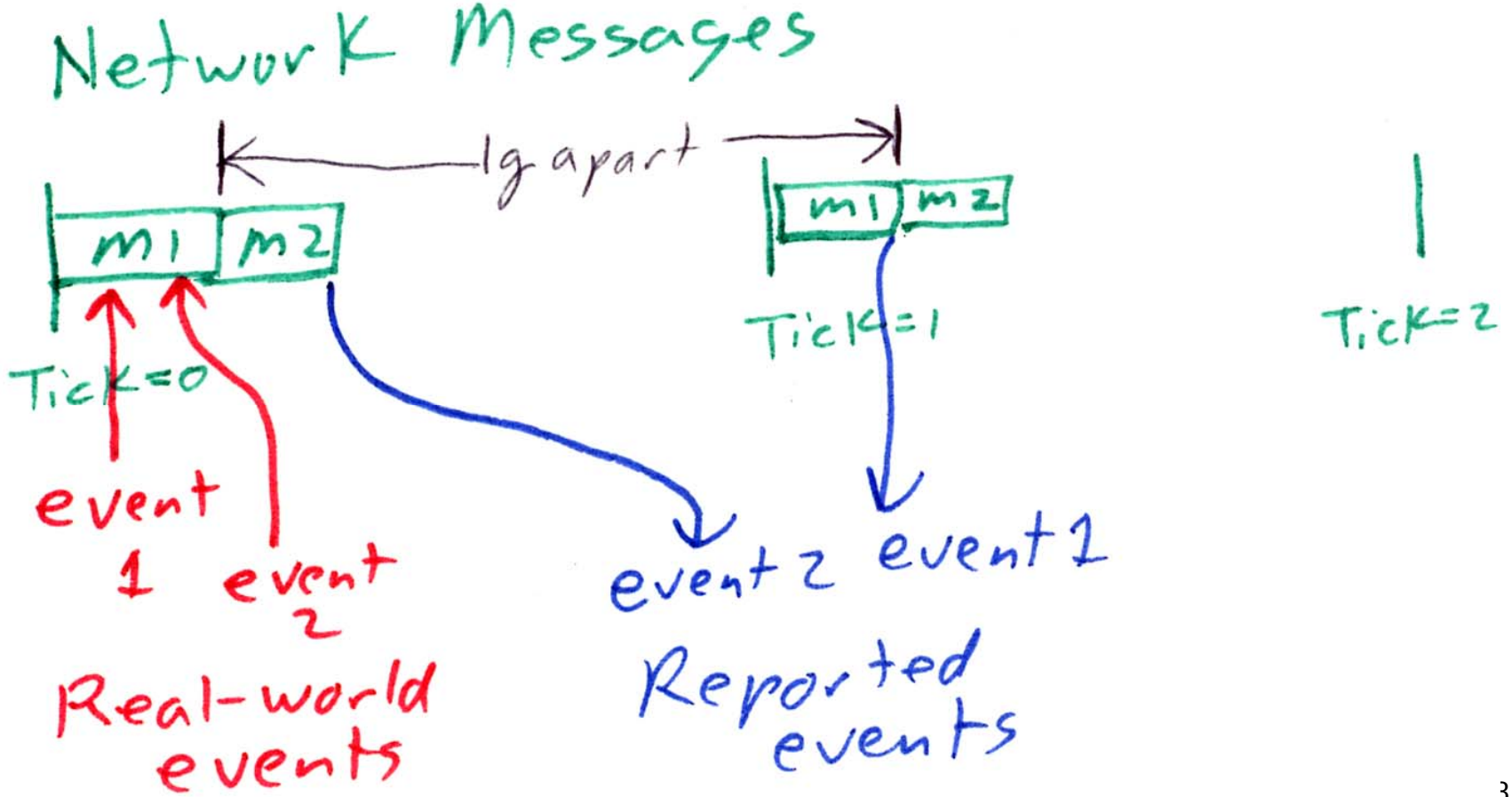
- **Assume order unknown (e.g., RMA scheduled Network or RTOS)**



- **In above example:**
  - Assume time-triggered messages sent somewhere within each global tick
  - Event 2 is reported by its message almost 2 ticks before Event 1
    - (If messages take "zero" time, events have to be >= 2 ticks apart to guarantee order)
  - Idea of "sparse time" – time stamps only increment at macro-tick boundaries

Need Only 1g Separation For Ordered Messages

- **If message order is *<u>always the same</u>* (e.g., TDMA network) only need 1g separation to guarantee unique order**
  - Any messages that arrive > 1g apart are guaranteed to be in correct order (assuming no lost/dropped messages)
  - But, this assumes you know network schedule, which might not be true



Distributed Time Measurement Errors

- **Offset**: difference in time at a particular instant per an omniscient observer
- **Precision**: ($\mathcal{S}$) maximum offset between any two clocks within system
- **Accuracy**: (A) offset between system time and the "real" time

Given this information:

"Real" correct time:	**14**
Node #1:	**18**
Node #2:	**13**
Node #3:	**17**

What is:

Offset: **Node 1 vs. Node 2 ____** **Node 3 vs. Node 2: ____**

Precision: **_____**

Accuracy: **Node 1 ____** **Node 2____** **Node 3___**

Clock Synchronization

- **Every once in a while, clocks must be reset to the "correct" time**
  - Consensus among nodes (improving precision)
  - Consensus with notional reference clock (improving accuracy)
- **State correction**
  - Agree on the time and fast-forward/rewind to that time
  - Simple, but introduces discontinuities in time base
- **Rate correction**
  - Speed up/slow down tick rate to converge to better time
  - More difficult to implement, less chance of a problem
  - GPS time is "rate steered"
    - GPS time is typically accurate within 200 ns to 1 microsecond
- **Fault Tolerant correction**
  - Usually, drop the lowest and highest clocks, then average the rest
  - (Advanced theory and correctness proofs apply here…)

Master Clock Synchronization

- **Master node says "it is now 1:32 PM"**
  - Assume that master node has access to high-quality time reference
  - Assume that master node never fails, or is redundant
  - Assume that master node messages have high priority

- **Embellishments:**
  - Use multiple round-trip messages to establish message latency; compensate time for message latency
  - Use a broadcast message instead of individual messages to each node
  - Use periodic broadcasts, and establish a local phase-locked-loop with master clock at each node

Distributed Time Synchronization

- **Improve precision by reducing variation**
  - Nodes vote to establish mean time value
  - Nodes adjust time to conform to mean
  - Only works well when:
    - Adjustments are tweaked to total zero in aggregate (to avoid system drift)
    - Node time drift is completely random (unbiased)

- **This is good enough if the time in the outside world doesn't matter**
  - If accuracy matters, use an external time base as one of the node times

  - Distributed time synch is used in most dependable embedded systems

g in an Embedded Network

- **Basic granularity limit is 2 t_{pd} in an embedded network**
  - Node A starts sending a bit
  - Node B "sees" a bit 1 t_{pd} later (t_{pd} = propagation delay)
    - Might have started sending a bit of its own during that t_{pd} interval
  - Node A "sees" results of potential interference from B yet another t_{pd} later

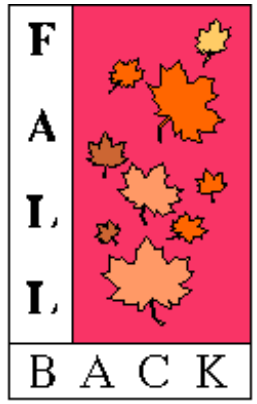
- **Consequences:**
  - Takes special care to achieve clock synchronization better than 2 t_{pd}
  - Bit times on some networks are limited to 2 t_{pd} in size to synchronize state machines controlling network protocol
  - And, of course, gate delays in network interface logic make it worse

- **Example:**
  - 1 Mbit CAN network; assume speed of signal in wire = 0.5 C
  - Maximum length: 2 t_{pd} = 1 psec length = 0.5 C * t_{pd} = 150 meters
    - But, tall buildings might be > 350 meters high... Have worse than 1 psec synch.

Daylight Savings Time & Time Zones

◆ Daylight savings time switches

- Which are declared annually by Congress and have been known to change
  - WW II had war-time daylight savings time to save energy
  - "Energy Crisis" in the 70's resulted in year-round daylight savings time
  - Only the Navajo nation within Arizona does DST (not the state; not the Hopi resv.)
- http://www.energy.ca.gov/daylightsaving.html
  - **Beginning in 2007**, Daylight Saving Time extended:
  - **2 a.m. on the Second Sunday in March to 2 a.m. on the First Sunday of November.**
  - This does not correspond to European dates!



www.time.gov

Users complain iPhone clock bungles time change

Instead of springing forward, some iPhone clocks fell back

Companies: Verizon Communications Inc. Com

Related Quotes

Symbol	Price	Change
VZ	35.85	-0.55

On Sunday March 13, 2011, 7:41 pm EDT

NEW YORK (AP) -- It's hard enough to get your bearings when the time changes twice a year. It's all but impossible when your phone starts playing tricks on you, too.

Users of Apple's iPhone peppered Twitter and blogs with complaints Sunday when their phones bungled the one-hour "spring forward" to daylight savings time that went into effect overnight Saturday.

One user complained of missing church, another of almost missing yoga. One called her iPhone stupid and several just asked for help.

It turns out some users' phones fell back one hour instead of springing forward, making the time displayed on the iPhone two hours off.

This is just the latest clock woe for Apple's chic iPhone. A clock glitch prevented alarms from sounding on New Year's Day, causing slumbering revelers to oversleep. The devices also struggled to adjust to the end of daylight savings time back in November.

The glitch affected iPhone owners who subscribe for phone service through AT&T and Verizon.

Apple, based in Cupertino, Calif., could not be reached for comment Sunday.

Problems With Time in the Real World

- **Coordinated Universal Time (UTC; the world time standard)**
  - Is not a continuous function due to leap seconds (and is only monotonic by putting 61 seconds in a minute just before midnight)
  - And, of course, leap year also causes discontinuities, although they're more predictable
- **Time zones**
  - Not just on hourly boundaries – Venezuela is UTC/GMT -4:30 hours; no DST
  - VCR auto-time-set might sync to channel from wrong time zone via cable feed
- **DST changeover date changes fairly often**
  - With little warning compared to a 10-20 year embedded system lifetime
- **"Y2K"**
  - The GPS 1024 week time rollover (a ship got lost at sea…)
  - And Unix rollover problem (January 19, 2038 03:15:07 GMT)
  - Leap year occurs more often … but still a problem

TechWeek europe

Enhancing business with technology – in association with eweek.com

Windows Azure Leap-Year Glitch Takes Down G-Cloud

Microsoft says that most services have now returned to normal after a day of chaos

On March 1, 2012 by Steve McCaskill 5

Microsoft has confirmed that a service outage that affected its cloud computing service Microsoft Azure, appears to be caused by a leap year bug.

The Government's G-Cloud CloudStore was among the sites affected by the outage, which Microsoft says has mostly been rectified.

Leap Year Bug

"Yesterday, 28 February, 2012 at 5:45 PM PST Windows Azure operations became aware of an issue impacting the compute service in a number of regions," wrote Bill Laing, corporate vice president of Server and Cloud at Azure in a blog post. "While final root cause analysis is in progress, this issue appears to be due to a time calculation that was incorrect for the leap year."

"Once we discovered the issue we immediately took steps to protect customer services that were already up and running, and began creating a fix for the issue," he explained. "The fix was successfully deployed to most of the Windows Azure sub-regions and we restored Windows Azure service availability to the majority of our customers and services by 2:57AM PST, 29 February."

Laing did concede however that some regions and customers were still experiencing issues and that as a result they may be experiencing a loss of application functionality.

"We are actively working to address these remaining issues," he added. "We sincerely apologise for any inconvenience this has caused. "

Government Issues

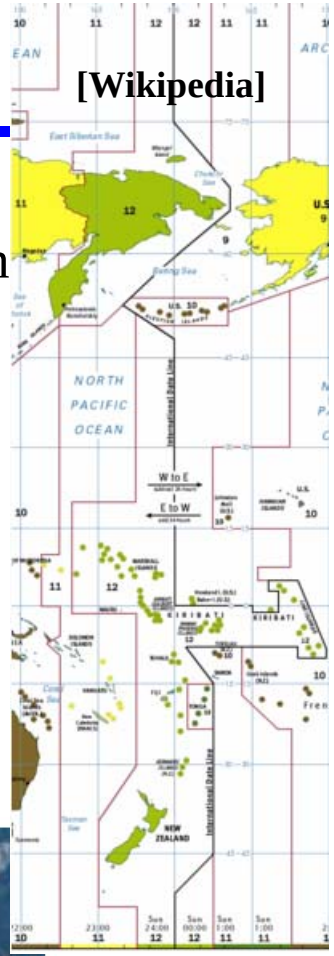
The Government's G-Cloud CloudStore, which was launched earlier this month, was taken offline due to the problems.

"Power outage on microsoft azure means #cloudstore is temporarily unavailable. Patch being applied so will update when normal service resumed," said a post on the official G-Cloud twitter account.

However a second message posted at 3:35pm GMT read: "Update on #cloudstore: microsoft are moving us to a different azure install and are confident we'll be up and running again by 4pm"

This is not the first time that Azure has gone offline. In March 2009, an outage left users unable to access the early test applications. This latest incident is unlikely to inspire confidence in IT managers still recovering from the Amazon Web Services (AWS) outage that occurred last April.

F-22 Raptor Date Line Incident



[Wikipedia]

- **February 2007**
  - A flight of six F-22 Raptor fighters attempts to deploy US to Japan
  - $360 million per aircraft (Perhaps $120M RE, rest is NRE)
  - Crossing the International Date Line, computers crash
    - No navigation
    - No communications
    - No fuel management
    - Almost everything gone!
    - Escorted to Hawaii by tankers
    - If weather had been bad, might have caused loss of aircraft
  - Cause: “It was a computer glitch in the millions of lines of code, somebody made an error in a couple lines of the code and everything goes.”


[DoD]

2013: NASA Declares End to Deep Impact Comet Mission

http://apod.nasa.gov/apod/image/0505/art1_deepimpact.jpg

Launched in 2005, the spacecraft memorably smashed a copper-jacketed probe into the comet Tempel 1 at 22,800 miles an hour (36,700 kilometers an hour) on July 4 of that year. It then flew through the debris cloud to capture the resultant fireworks, the first close inspection of a comet's interior. (See "'Deep Impact' Comet Revealed by NASA Flyby.")

The $267 million spacecraft later flew by the comet Hartley 2 in 2010, and this year it captured images of comet ISON, which is headed toward a close encounter with the sun in November.

But now the Deep Impact spacecraft appears to be lost.

Mission controllers last radioed the spacecraft on August 8, after which communications were lost, according to a statement from the Jet Propulsion Laboratory in Pasadena, California. After a month of attempts to restore communications through the NASA Deep Space Network, the controllers have declared the mission "lost," concluding that a computer glitch likely doomed the spacecraft.

"Basically, it was a Y2K problem, where some software didn't roll over the calendar date correctly," said A'Hearn. The spacecraft's fault-protection software (ironically enough) would have misread any date after August 11, 2013, he said, triggering an endless series of computer reboots aboard Deep Impact.

Dan Vergano
National Geographic
Published September 20, 2013

http://news.nationalgeographic.com/news/2013/09/130920-deep-impact-ends-comet-mission-nasa-jpl/

Review

- **Distributed time**
  - When things happen; chain of causal events
  - Relating messages to causality

- **Clocks and time ticks**
  - It is fundamentally impossible for all nodes to have exactly the same clock time
  - Limitations of network messaging affect clock synchronization
    (if you use the same network to distribute time and announce events, it is difficult to have a time base more precise than event announcement jitter)

- **Clock synchronization approaches**
  - Tradeoff of changing rate of change or changing value