Plausible Clocks with Bounded Inaccuracy

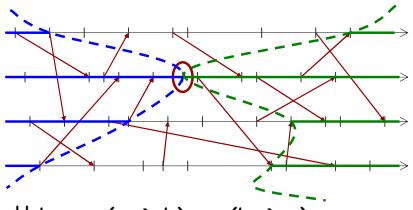
Brad Moore, Paul Sivilotti

Computer Science & Engineering
The Ohio State University

paolo@cse.ohio-state.edu

Causality & Concurrency

- a → b ⇔ exists a path from a to b



- a || b ⇔ ¬(a → b) ∧ ¬(b → a)

Time-Stamping Systems

- A TSS is an algorithm for assigning
  - "time stamps" to events, T(a)
  - "time tags" to messages
- Implementing a TSS requires
  - STAMP( )
    - Input: previous local stamp, incoming tag
    - Output: new local stamp
  - TAG( )
    - Input: previous local stamp
    - Output: tag to add to outgoing message
  - COMP( )
    - Input: two time stamps
    - Ouput: one of { <, =, || }

Example TSS: Lamport Clocks

- Time stamps are integers
- Time tags are integers
- STAMP:
  - for local and send events, increase stamp
  - for receive events, use max of stamp & tag
- TAG:
  - use local time stamp to tag messages
- COMP:
  - < and =, integer comparison
  - || when same value but on different processes

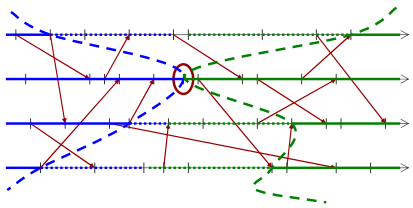
Example TSS: Vector Clocks

- Time stamps are arrays of N integers
- Time tags are arrays of N integers
- STAMP:
  - for local and send events, increase local element
  - for receive events, use element-wise max of stamp & tag
- TAG:
  - use local time stamp to tag messages
- COMP:
  - < when all elements are <= and at least one is <
  - = when all elements are =
  - || when some element is < and another is >

Plausibility

- Weak clock condition (WCC)
  - a → b ⇒ T(a) < T(b)
- A TSS satisfying the WCC is plausible iff it is consistent with underlying hb partial order
  - every pair ordered by TSS is causally related
  - no distinct events are given equal stamps
- Example: Lamport clocks
  - orders (almost) all pairs!
    - exception: same value on different processes
  - every pair identified as concurrent, is indeed concurrent in hb partial order
  - stamps from distinct events can always be distinguished (add process id to stamp)

Ordering from Lamport Clocks



Application of Plausibility

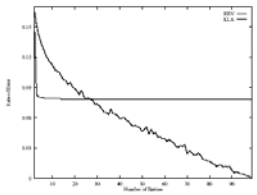
- Correctly identifying causally related pairs generally necessary for safety
  - arbitration (resource allocation, mutex)
  - consistent serialization (cache coherence)
  - playback for distributed debugging
- Correctly identifying concurrent pairs generally important for performance
  - consistent cuts require concurrent sets
  - cache consistency and invalidation protocols
  - snapshots (deadlock & termination detection, checkpoints for rollback)

Inaccuracy of a Plausible TSS

- Two kinds of errors:
  - "false concurrency": related events stamped concurrent
    - WCC means a plausible TSS makes of none of these errors
  - "false ordering": concurrent events stamped as ordered
    - plausible clocks may make this error
- "Inaccuracy" measures how many such mistakes (false orderings) are made in a given run
- Fundamental result:
  - perfect accuracy requires O(N) message overhead
  - (so Vector clocks are optimal)
  - does not scale to large systems
- Motivates natural research question:
  - *Can we get good expected-case accuracy with less message overhead?*

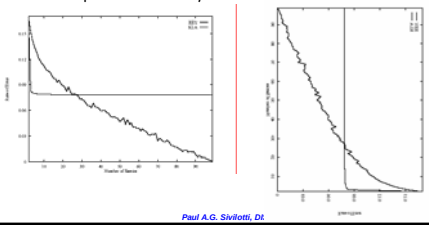
Designing a Plausible TSS

- Previous approaches:
  - Fix time tag size (ie message overhead)
    - constant size
  - Measure resulting accuracy through simulation
    - expected case analysis



Inversion of Design

- Our approach:
  - Fix inaccuracy
    - some constant upper bound
  - Measure resulting message complexity through simulation
    - expected case analysis

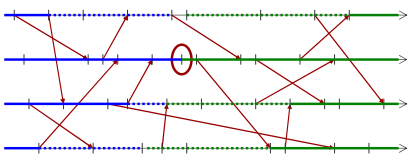


Towards a New Metric: Error Count (δ)

- *Error count* for an event is the number of false (pre) orderings, for that event

$$\delta(P,H,b) = \left| \left\{ a \in H :: a \,||\, b \wedge a \overset{P}{\rightarrow} b \right\} \right|$$

  - i.e., number of *blue* dots

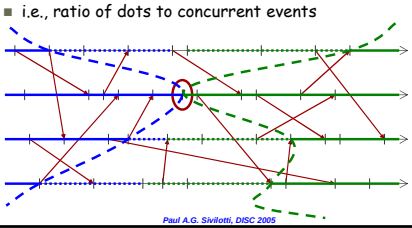


Towards a New Metric: Inaccuracy (ρ)

- *Inaccuracy* is the ratio of false orderings, averaged over entire run

$$\rho(P,H) = \frac{2 \times \left(\sum b \in H :: \delta(P,H,b)\right)}{\left|\{a,b \in H :: a \parallel b\}\right|}$$

  - i.e., ratio of dots to concurrent events

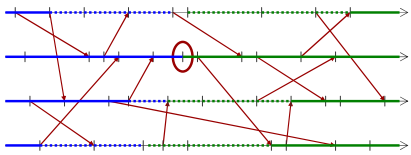


Introducing a New Metric: Imprecision (ψ)

- *Imprecision* is the max number of false (pre) orderings for a given time stamp

$$\psi(P,s) = \left(\text{Max } H \in \mathbf{H}(P,s), a \in H : P(a) = s : \delta(P,H,a)\right)$$

  - i.e., worst-case number of blue dots
  - independent of particular run (property of a time stamp)

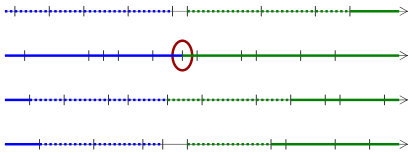


Introducing a New Metric: Imprecision (ψ)

- *Imprecision* is the max number of false (pre) orderings for a given time stamp

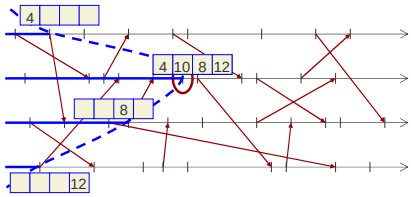
$$\psi(P,s) = \left(\text{Max } H \in \mathbf{H}(P,s), a \in H : P(a) = s : \delta(P,H,a)\right)$$

  - i.e., worst-case number of blue dots
  - independent of particular run (property of a time stamp)



Algorithmic Approach

- Recall Vector Clocks
  - stamp is an array of values, V[1..N]
  - V[i] is "most recent event" on process i



Compressing Time Tags

- If multiple entries have the same value, they can share an entry in vector
  - problem: unlikely to have exact equality
- Use a range for vector entry
  - most recent event guaranteed to be in range
  - the larger the range, the greater the imprecision
- Allow ranges to grow/shrink (and number of vector entries to increase/decrease)
  - maintain constant precision
  - message overhead may vary over the run

Three Claims

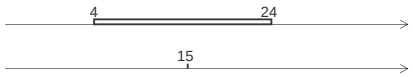
1. The algorithm is a correct plausible TSS
   - satisfies the WCC (and distinguishes events)
2. The imprecision can be controlled
   - different sources of imprecision: local stamps and tagged messages
   - their combination does not lose too much information
3. The algorithm achieves good performance
   - measured in terms of tag size
   - expected case evaluation

Outline of the Talk

- Background
  - model, causality (→), concurrency, TSS
- Plausible TSS
  - WCC, accuracy vs precision
- Algorithmic Approach
  - use ranges instead of precise values
- Algorithm Description
  - time intervals
  - implementation of STAMP, TAG, COMP
- Justification of claims
  1) correctness (algorithm is a plausible TSS)
  2) imprecision is bounded
  3) performance is reasonable

Time Intervals

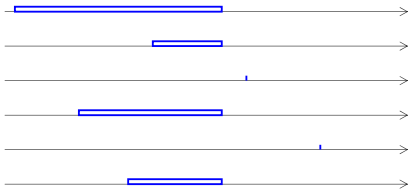
- Interval given by beginning and end points
  - if these are the same, interval is "precise"



- An interval is associated with each process
  - goal: the "local time" of the most recent causally related event is within the interval
  - interval associated with *this* process is precise
  - local storage is linear in N

Time Stamps

- A time stamp is a vector of intervals
- Satisfies two invariants:
  A) imprecise intervals share the same end point
  B) precise intervals are all greater than this value

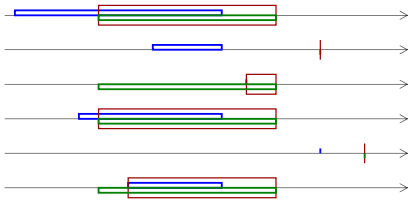


Time Tags

- Subset of time stamps
  - a vector of intervals, with properties A & B
- Satisfies an additional invariant:
  C) imprecise intervals share the same begin point
  - all imprecise intervals are the same (a "bucket")

Implementation: STAMP

- For local/send events: increase local (precise) interval
- For receive events (stamp + tag)
  - each interval in new stamp defined by:
    - begin = max of begin points
    - end = max of end points
  - increase local (precise) interval



Implementation: TAG

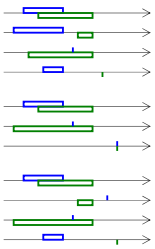
- Given a time stamp and an imprecision budget:
  - build smallest acceptable tag (ie use as big a bucket as possible)
- Algorithm:
  - start with everything in bucket
  - repeat: move most recent interval from bucket to its own entry
  - until bucket imprecision is within budget

Implementation: COMP

- Comparing individual intervals (i,j)
  - i = j : same precise intervals
  - i < j : no overlap, end of i < beg of j
  - i ≈ j : some overlap
  - i <≈ j : ¬(j < i)

Implementation: COMP

- Comparing vectors of intervals (s,t)
  - s < t
    - all i in s, j in t :: i <≈ j
    - exists an i in s, j in t :: i < j
  - s = t
    - all i in s, j in t :: i ≈ j
  - s || t
    - exists an i in s, j in t :: i < j
    - exists an i in s, j in t :: j < i

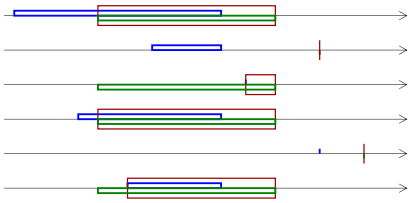


Justifying the Claims

1. The algorithm is a correct plausible TSS
   i. stamp and tag invariants [A-C]
   ii. a → b ⇒ T(a) < T(b) [WCC]
   iii. a = b ⇔ T(a) = T(b) [no 2 equal stamps]
2. The imprecision is bounded (worst case)
3. The algorithm achieves good performance

Claim 1: Correctness

- Part (i): stamp and tag invariants
  - imprecise intervals share an end point
  - precise intervals are all greater than this value
- Proof intuition
  - new shared end point = max of old shared end points

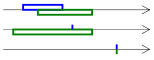


Claim 1: Correctness

- Part (ii): WCC holds
  - a → b ⇒ T(a) < T(b)
- Surprising complication: <≈ is not transitive
  - i <≈ j
  - j <≈ k
  - ¬(i <≈ k)
- Proof intuition
  - along every actual chain, time stamps are non decreasing
  - every happens-before pair is joined by a chain

Claim 1: Correctness

- Part (iii): No two equal stamps
  - a = b ⇔ T(a) = T(b)
- Surprising complication:
  - stamps are equal when all entries overlap



- Proof intuition:
  - local interval is always precise
  - for events on same process, the local interval is unique
  - for events on different processes, both local intervals can not be identical
    - equality of local interval implies causality
    - hence, equality of both implies a cycle in causality

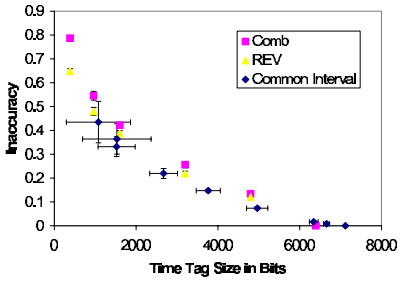
Claim 2: Guaranteed Bound

- Surprising since:
  - stamp intervals can increase on receive!
  - tagging increases imprecision
- Key observation:
  - after receive, each stamp interval smaller than either old stamp or tag
- Important properties:
  - ψ(stamp) <= max(ψ(old stamp), ψ(tag))
  - ψ(tag) <= BOUND
- Result:
  - stable.ψ(stamp) <= BOUND

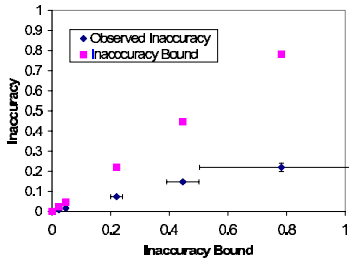
Claim 3: Good Performance

- Comparison challenges
  - trade-off is similar (message size vs errors)
  - but imprecision is not accuracy!
- Simulation results
  - varying number of processes
  - fixed topology
    - some high degree nodes some low
  - varying intercommunication delays

Performance: Good News (Inaccuracy vs. Message Size)



Performance: Good (?) News (Observed vs Bound)



Conclusions

- Plausible clocks
  - provide tradeoff between message size and accuracy
- Contribution: Imprecision metric
  - property of a time stamp (run independent)
  - worst-case behavior
  - unbounded for any constant message size
- Contribution: Bounded imprecision plausible clocks
  - time stamps have intervals (messages have buckets)
  - message overhead grows and shrinks as necessary
  - guarantee on amount of imprecision (hence inaccuracy)
- Contribution: Promising performance
  - actual accuracy generally better than imprecision bound

Plausible Clocks with Bounded Inaccuracy

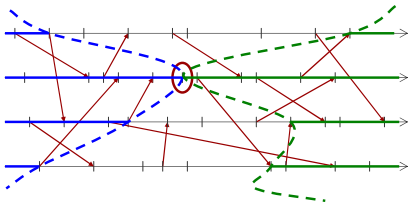
Brad Moore, *Paul Sivilotti*

Computer Science & Engineering
The Ohio State University

paolo@cse.ohio-state.edu

Bounding Inaccuracy

- Concurrency ratio, ε: $\varepsilon(H) = \frac{1}{2} \times \frac{\left|\{a,b \in H :: a \parallel b\}\right|}{|H|}$
  - i.e., (avg) number of events in concurrent window



  - For "well-behaved" computations, this is constant

Bounding Inaccuracy (Cont'd)

- Rewrite inaccuracy using ε(H)

$$\rho(P,H) = \frac{2 \times \left(\sum b \in H :: \delta(P,H,b)\right)}{\left|\{a,b \in H :: a \parallel b\}\right|}$$
$$\leq \frac{2 \times \left(\sum b \in H :: \psi(P, P.stamp(b))\right)}{\left|\{a,b \in H :: a \parallel b\}\right|}$$
$$= \frac{1}{\varepsilon(H)} \times \frac{\left(\sum b \in H :: \psi(P, P.stamp(b))\right)}{|H|}$$
$$\leq \frac{1}{\varepsilon(H)} \times BOUND$$

Performance: Bucket Size