MITRE Technology Program

Information Management

Application QoS Based Time-Critical Automated Resource Management in Dynamic BM/C^2 Systems

E. Douglas Jensen

The MITRE Corporation

jensen@[mitre | real-time].org
http://www.[real-time | mitre].org

Revised 31 August 04

MITRE

Summary

- This project is researching a novel paradigm for expressing, enforcing, and formally reasoning about time-criticality of machine-to-machine resource management in battle management (BM) and C^2 systems
- Such systems are largely dynamic and asynchronous, and have time-critical actions in the $O(10^{-1} – 10^{3})$ seconds
- Thus they fall into a neglected gap between traditional static periodic "real-time" systems, and traditional "any time" scheduling/planning systems (e.g., for logistics)
- The paradigm uses application-level QoS (AQoS) metrics (such as track quality, circular error probable, etc.) to derive *time/utility functions* (TUF's) for completing tasks,

  and then manages resources to maximize *accrued utility*
- This paradigm has been successfully employed in two unclassified experimental BM/C^2 demonstration systems
- It is the topic of collaboration between MITRE and academia

Many important time-critical systems (such as for BM/C^2) have significant dynamic actions

- Many important time-critical control systems do not fit the "real-time" stereotype of
  - small scale
  - static
  - periodic
  - centralized
  - performing monitoring and control of simple devices
  - time frames in the microsecond and millisecond range

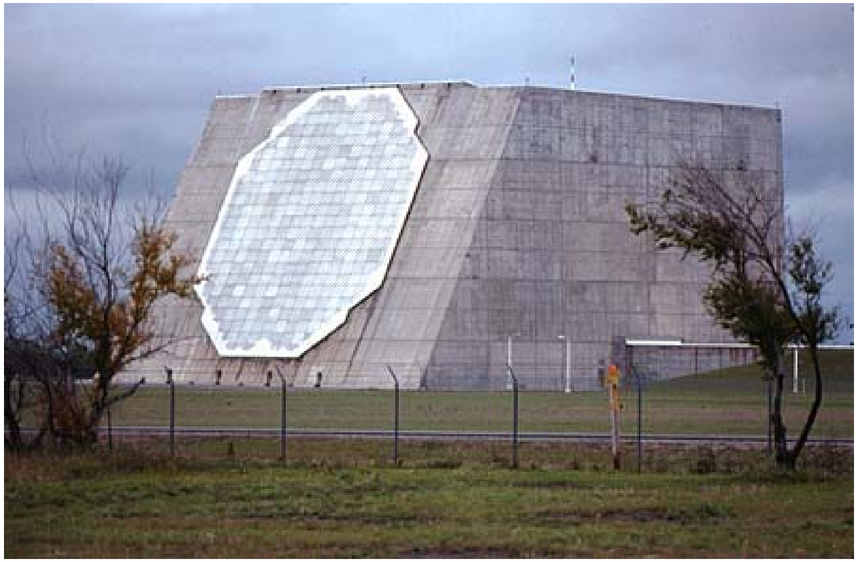
- Instead, they are
  - large scale in various dimensions
  - dynamic
  - "mesosynchronous"
  - distributed
  - performing closed loop machine-to-machine control at any level(s) of an enterprise
  - operating in the second to minutes time frame

Priorities are widely used in time-critical systems, but they have major disadvantages

- Priority assignments are not modular – they require global knowledge of all other priority assignments (whereas time constraints, such as deadlines, do not)
- Semantics of priorities can denote any or all of: urgency, relative importance, execution precedence
- Priorities are assigned by users, system and application software, and hardware
- Managing the assignments and changing of priorities is one of the most notoriously difficult and time-consuming activities in the life cycle of computing systems

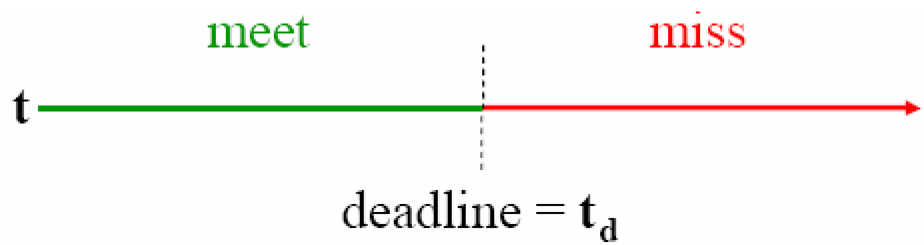
Historical background: in 1977 Jensen created a novel scheduling paradigm for dynamic time-critical systems

- U.S. Army Safeguard Command's AN/FPQ-16 Perimeter Acquisition Radar Characterization System
- Located at what is now the USAF Space Command's Cavalier Air Force Station, in Cavalier, North Dakota
- Initially poor radar performance
- Simulated performance with the new scheduling paradigm was deemed to possibly violate SALT II, so paradigm not deployed

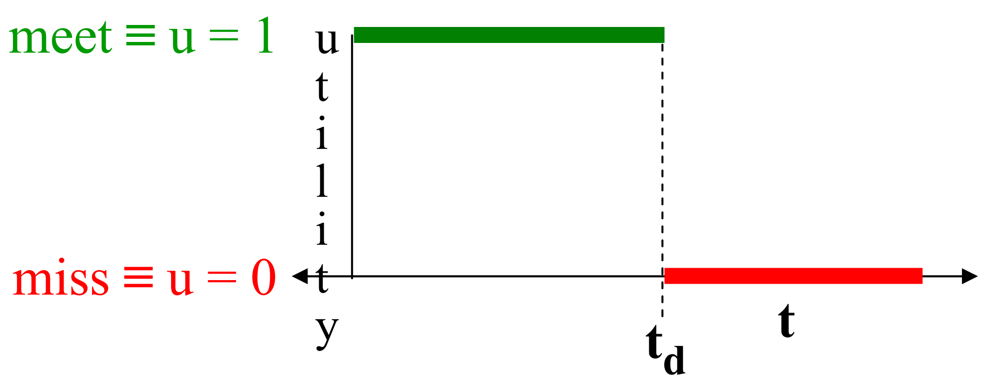


Hard deadlines and general deadlines can be represented by utility as a function of time

- **Deadlines, as popularly spoken of in "hard" real-time computing, are only binary: an action either meets or misses it**



- **As a function, a hard deadline is a unit-valued downward step, and a general deadline may have any two values**

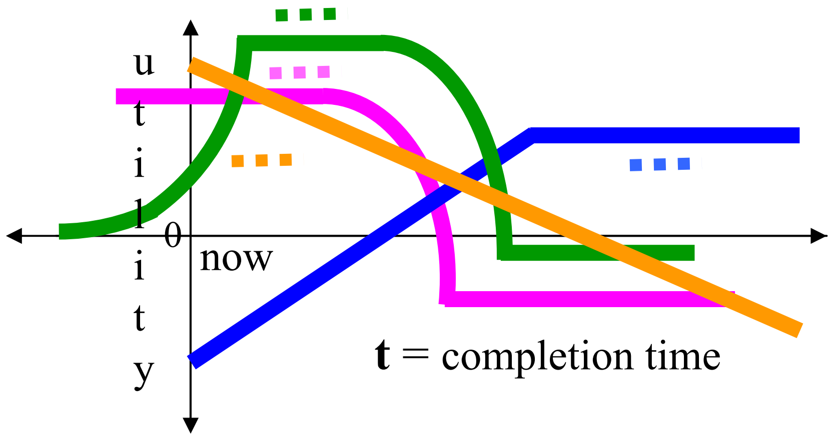


- **But in most real world cases, *lateness* is the actual criterion**

The two keystone concepts in our paradigm are time/utility functions and utility accrual scheduling

- Time/utility functions (TUF's)
  - express the utility to the system (derived from AQoS metrics) of completing an activity (e.g., service) as an application- or situation-specific function of when it completes

Example

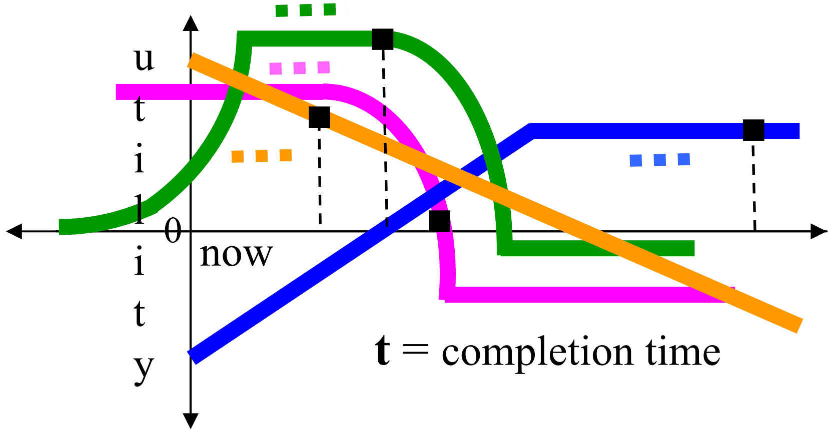


General time/utility functions

The two keystone concepts in our paradigm are time/utility functions and utility accrual scheduling

- Time/utility functions (TUF's)
  - express the utility to the system (derived from AQoS) of completing an activity (e.g., service) as an application- or situation-specific function of when it completes
- Utility accrual (UA) scheduling algorithms
  - schedule activities according to optimality criteria based on
    - accruing utility – such as maximizing the sum of the utilities
    - satisfying dependencies such as resource constraints, etc.

Example



General time/utility functions

Expected or max execution time
Example scheduled completion times

Schedule to maximize
$U = \sum u_i$ ■

Subsequently, we have been conducting research to advance and apply this paradigm

- From then until now, Jensen and his colleagues have continued to develop and refine the paradigm
- In CMU's CS department, two of Prof. Jensen's Ph.D. students – Locke, Clark – wrote their theses on this topic
  - devised new scheduling algorithms
  - proved theorems about them
  - simulated the algorithms
  - implemented the algorithms in our Alpha real-time distributed OS kernel
- The Open Software Foundation (now The Open Group) introduced Alpha features into the MK7 microkernel (a Mach 3 descendant) on which they built the OSF/1 OS
  - Wells, formerly the Alpha Release 2 Program Manager
  - Clark
  - others

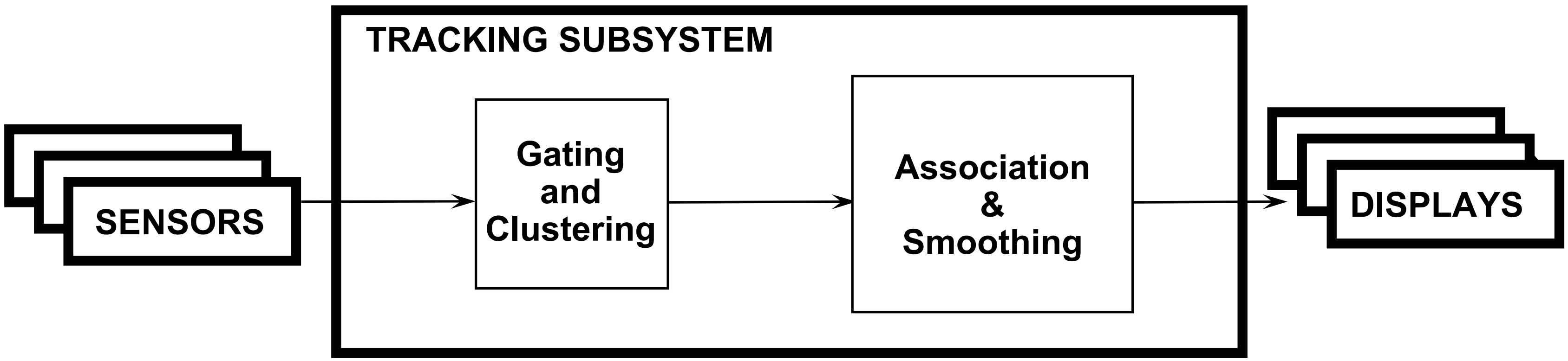
Worked examples using TUF/UA scheduling

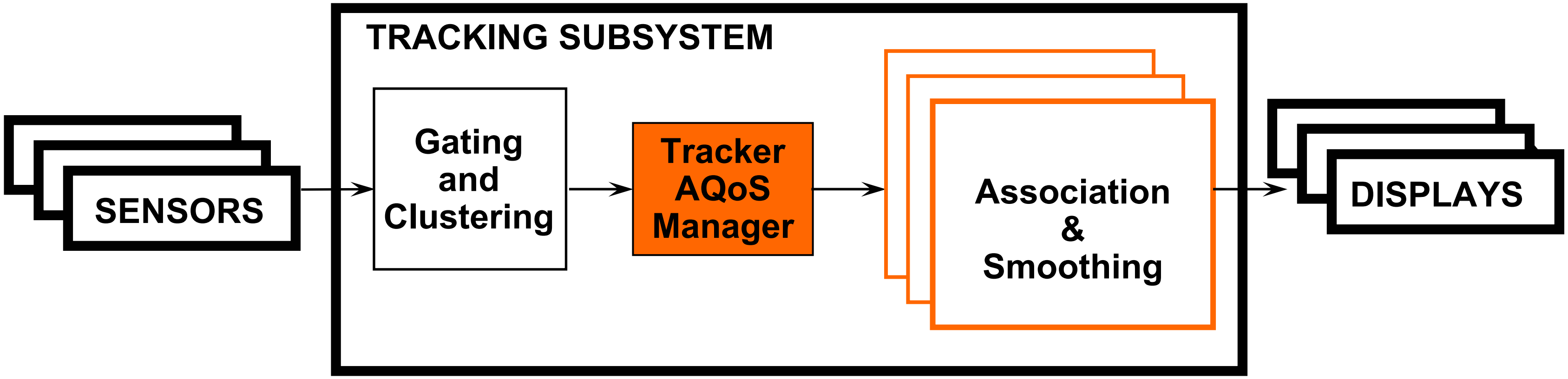
- This paradigm has been applied in two significant unclassified BM/C² demonstration systems
- Next we briefly illustrate the paradigm in the context of an air surveillance tracking application (papers, slides, and tech report available)
- Then we show one facet of the paradigm's use – not employed in the surveillance tracker – in a notional coastal air defense application (papers, slides, and tech report available)
- The paradigm is employed in a large complex demonstration application that is currently being constructed but that cannot be discussed here

TUF/UA sequencing paradigm worked example 1: AWACS air surveillance mode tracker

- MITRE (with collaboration from the Open Group) applied our paradigm in a demonstration AWACS system
- Implemented the AWACS air surveillance mission
- It is easy and common for there to be so many sensor reports that the system becomes computationally overloaded, which causes sectors of the sky to “go blank”
- Currently, operators have knowledge-intensive manual work-arounds for certain overload situations
- Our objective was to improve graceful overload handling by automatically
  - applying the right computational resources
  - to the right tracks
  - at the right times

Tracking system: original and adaptive

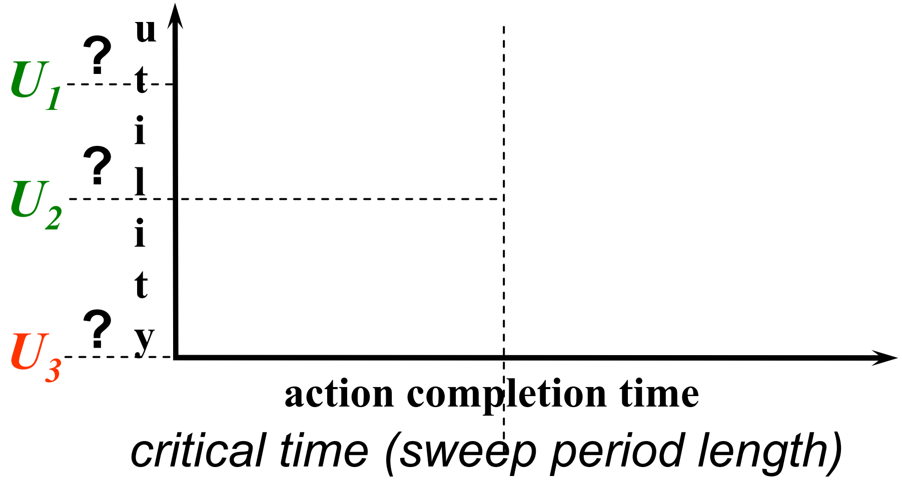




The AWACS sensor properties imply a general utility function for the association computation

- Association is the most computationally demanding part of tracking, so we focus on that in this presentation
- There are two sensors (radar and IFF) sweeping 180º out of phase with a 10-second period,

  which suggests the TUF has
  - a "critical time" at the 10-second period length
  - at least two distinct non-zero utilities before the critical time
  - a third distinct, lower, utility after the critical time



Determine the thread TUF shape prior to the critical time

- **Prior to the critical time**
  - processing a sensor report for one of these tracks in under five seconds (half the sweep period) would provide better data for the corresponding report from the out-of-phase sensor

    so the utility decreases with time
  - the TUF had to decrease linearly due to an implementation artifact in this experimental system –

    the OS (OSF/RI's MK7.3A) TUF scheduling algorithm allowed only one critical time
  - the slope was derived empirically

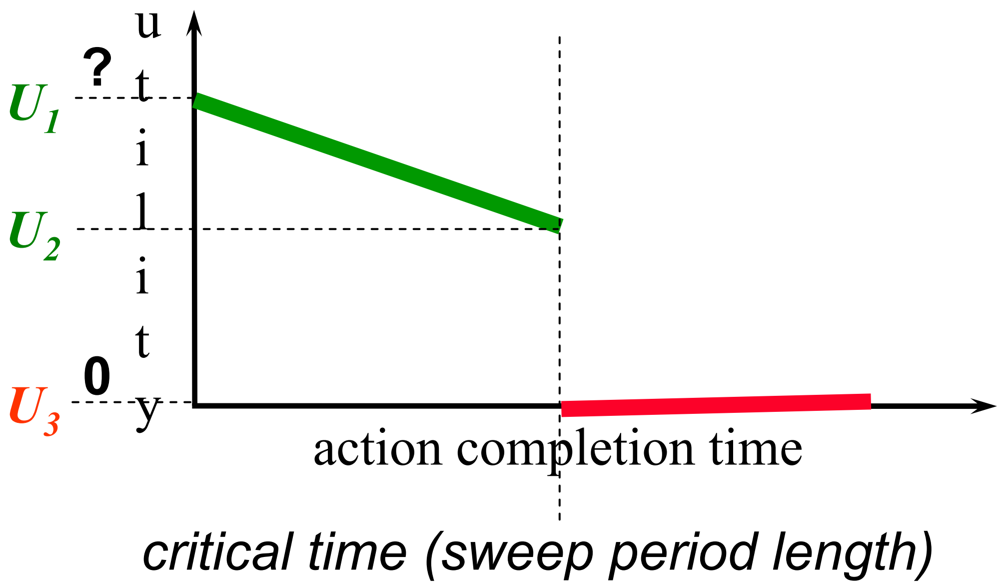
Determine the thread TUF shape after the critical time

- After the critical time
  - utility is zero, because newer sensor data has probably arrived
  - if the processing load in one sensor sweep period is so heavy that it couldn't be completed,

    probably the load will be about same in next period, so there will be no capacity to also process data from the previous sweep
  - a tracker that could process older as well as current data would
    - be significantly more complex
    - probably delay the track update

That established the TUF shape for the tracker's association threads

- A critical time at the sweep period length
- Linearly decreasing utility until the critical time
- Zero utility after the critical time
- Next, the utility value U_1 had to be determined



Tracker domain experts' preferences in terms of track QoS metrics imply the thread utility values

- Don't drop tracks, because they are expensive to re-create
- User-identified "important" tracks receive preference
- User-identified "important" geographic regions receive preference
- Maneuvering tracks need to be updated more frequently than non-maneuvering tracks
- Potentially high threat tracks receive preference
- High speed tracks receive preference
- Tracks with poor state estimates receive preference

Three application-level QoS metrics for an AWACS surveillance tracking application were chosen

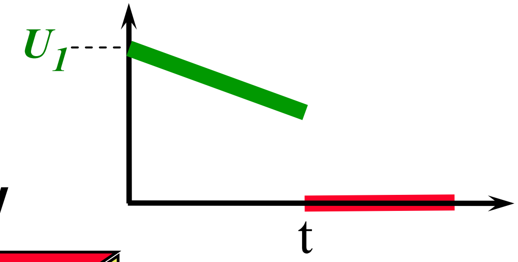
- ❑ **Quality – 0 to 7**
  - traditional measure of the amount of recent sensor data incorporated in a track record
  - incremented or decremented after each radar scan
- ❑ **Accuracy – "high" or "low"**
  - a measure of the uncertainty of the estimate of a track's position and velocity
  - derived from traditional Kalman filter processing
- ❑ **Importance – "high" or "low"**
  - traditionally, operator-identified based on geography, threat, and other characteristics

We established 12 combinations of track AQoS metrics

- Track state poor
- Track state marginal
- Track state OK

High/Low Importance

Track Accuracy



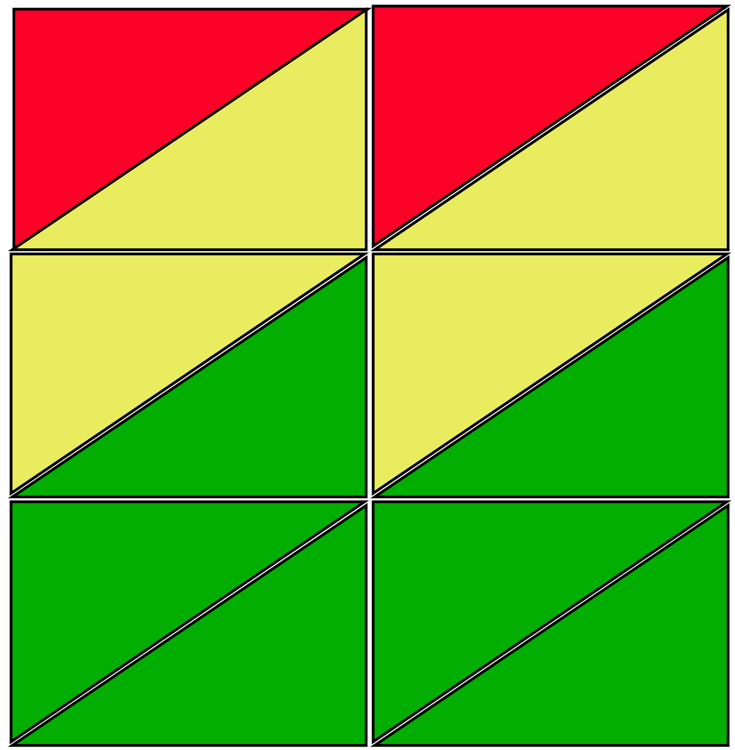
High | Low

Track Quality

Low 1 - 2

Medium 3 - 4

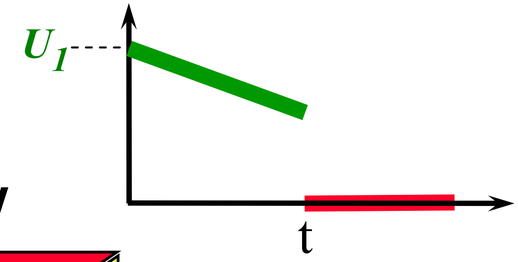
High 5 - 7



What are the relative utilities of these 12 cases of tracks?

The initial utility U_1 of an association for a track report is derived from track AQoS metrics by gedanken experiments

- Track state poor (red)
- Track state marginal (yellow)
- Track state OK (green)



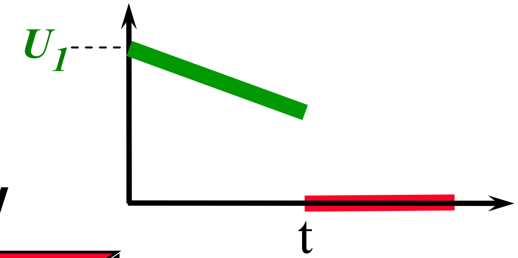
High/Low Importance

Track Quality		Track Accuracy: High	Track Accuracy: Low
Low	1 - 2	5500 / 910	6000 / 1000
Medium	3 - 4	700 / 30	800 / 40
High	5 - 7	53 / 10	65 / 20

Domain experts judgment on the relative utilities of these 12 cases of tracks

The initial utility U_1 of an association for a track report is derived from track AQoS metrics by gedanken experiments

- Track state poor
- Track state marginal
- Track state OK



Track Accuracy

Track Quality	High/Low Importance	High	Low
	Low 1 - 2	5500 / 910	6000 / 1000
	Medium 3 - 4	700 / 30	800 / 40
	High 5 - 7	53 / 10	65 / 20

E.g., completing an association for a high importance, low accuracy, low quality track yields 600 times more utility than for a low importance, high quality, high accuracy track

Domain experts judgment on the relative utilities of these 12 cases of tracks

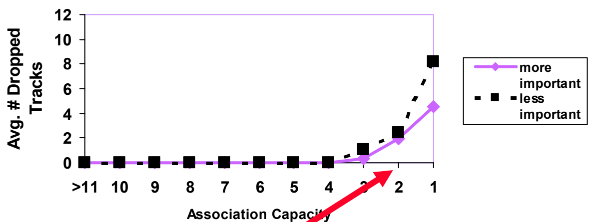
The association (and other) threads are scheduled based on their utility functions

- For the association threads, the tracking application selects the established TUF from the OS scheduler's library of shapes
- The tracking application does a look-up in the utility U_1 table for each association thread before calling the OS scheduler
- A utility-based processor-scheduling policy in the OS schedules threads according to a heuristic that attempts to maximize total accrued (in this case, summed) utility

Utility-based scheduling provided better AQoS than traditional FIFO and priority scheduling

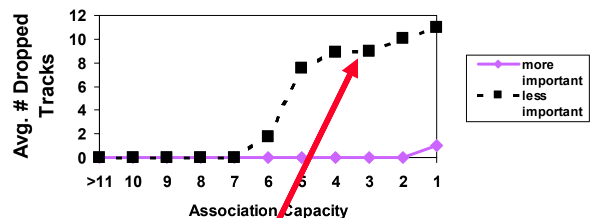
FIFO

Avg. # Dropped Tracks versus Association Capacity For FIFO Priority



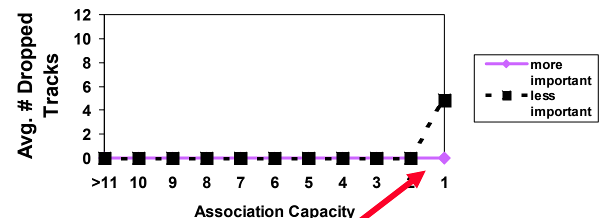
Priority

Avg. # Dropped Tracks versus Association Capacity For Fixed Priority



Utility-Based

Avg. # Dropped Tracks versus Association Capacity For Dynamic Priority

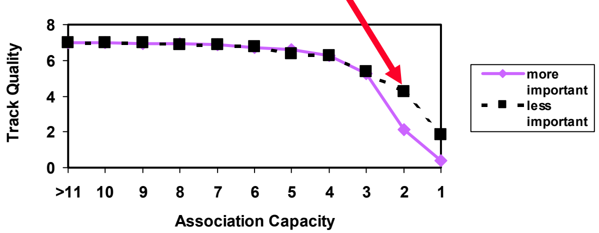


High Priority Tracks Were Dropped and have Bad TQ

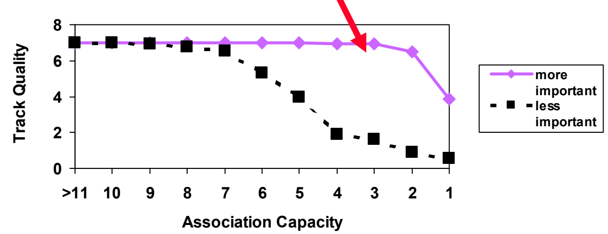
Drop *Low Priority* Tracks to Get Better TQ on *High Priority* Tracks

No *High Priority* Tracks Dropped. Overall Better TQ

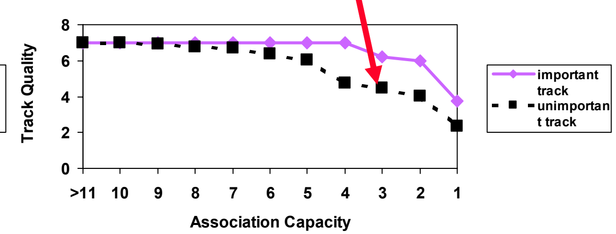
Track Quality versus Association Capacity For FIFO Priority



Track Quality versus Association Capacity For Fixed Priority

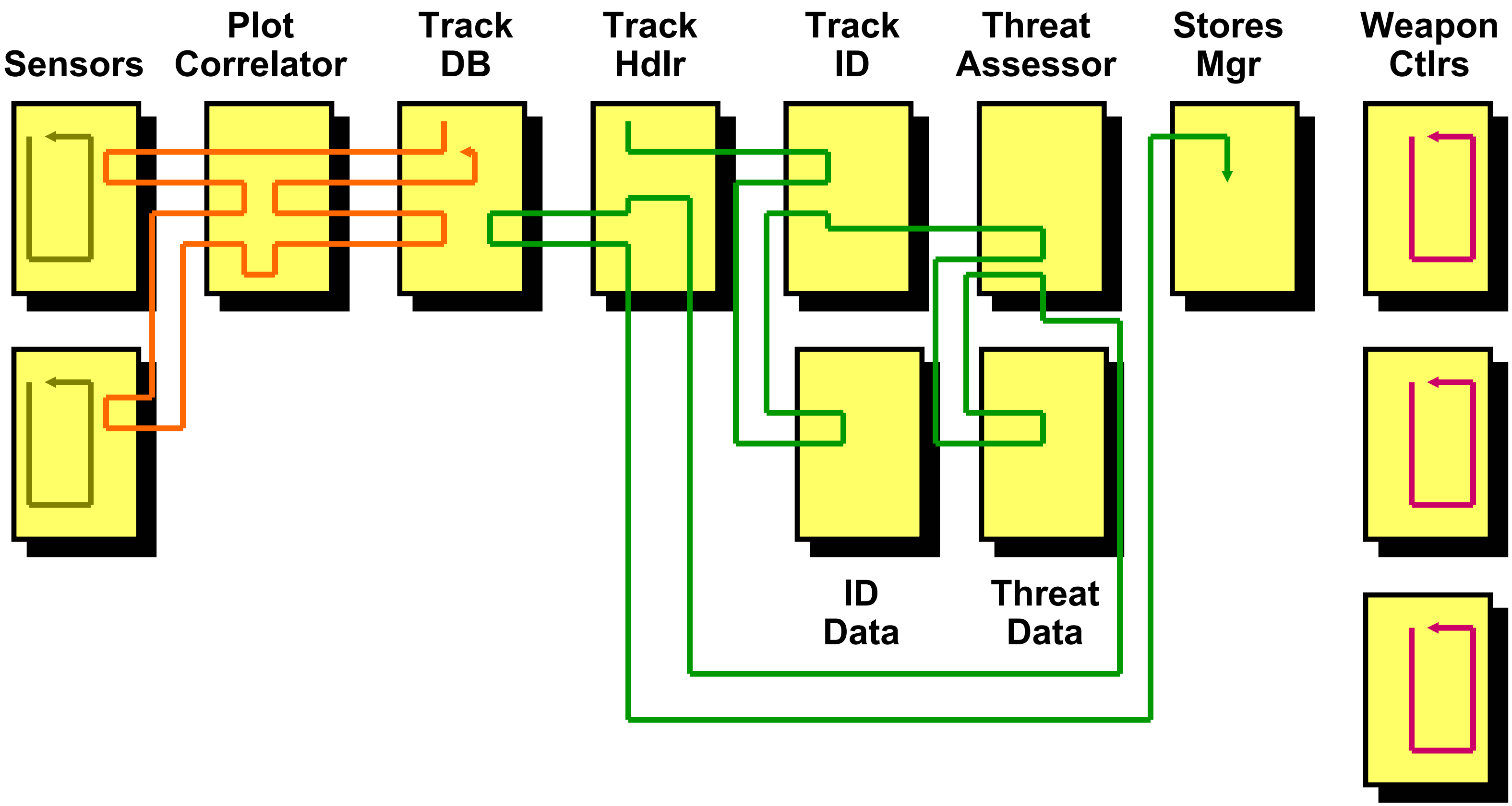


Track Quality versus Association Capacity For Dynamic Priority



Key:
Track Quality: 0-7 (7 = Ideal)
Association Capacity = # Tracks Processed under Constraint

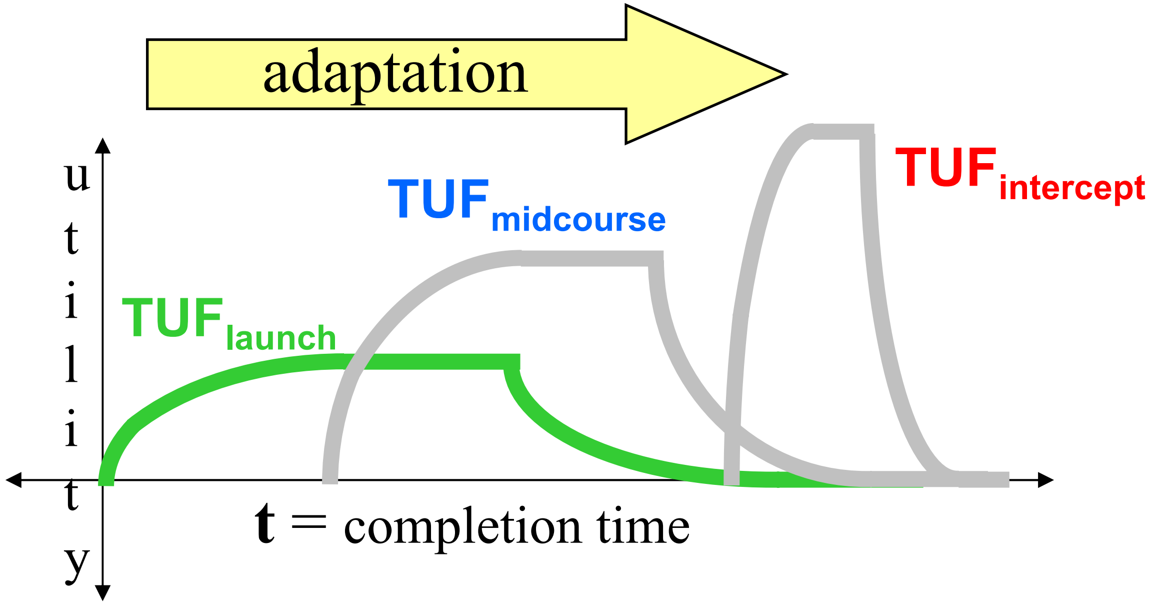
TUF/UA sequencing paradigm worked example 2: coastal air defense with guided interceptors



***Distributable Threads*: a programming model for end-to-end timeliness in distributed systems – created by Jensen's CMU Alpha OS team and now in Real-Time CORBA 1.2 (née 2.0)**

The timeliness requirements for the interceptor missile control threads vary over the course of an engagement

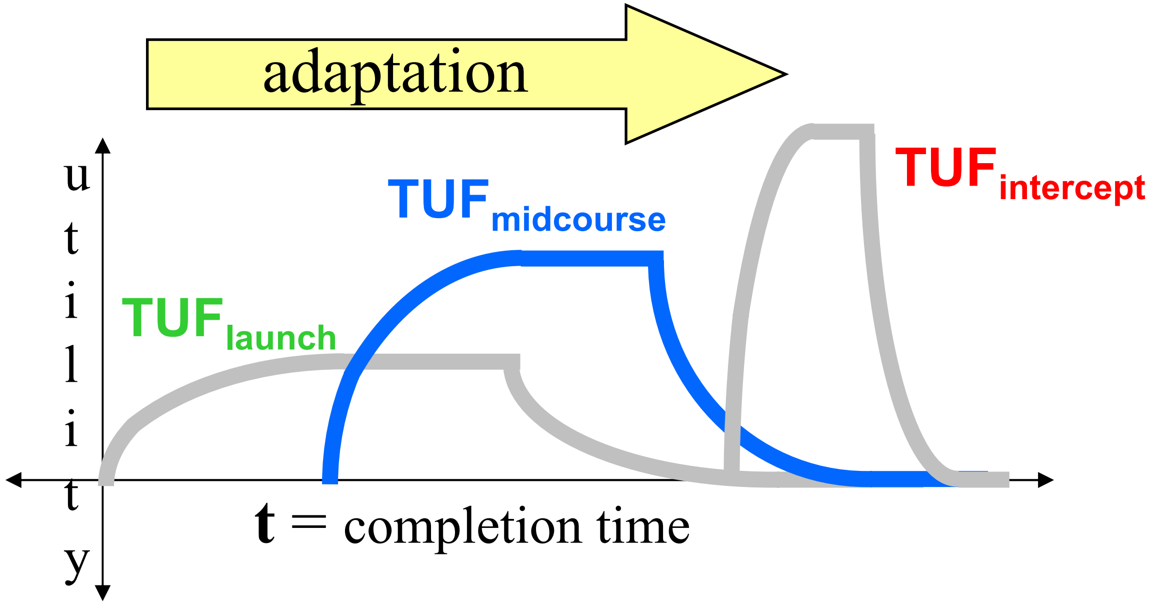
- **After launch of the interceptor, the guidance control threads must issue timely repetitive course updates to ensure a successful intercept**
- **The required timeliness of these updates, and the importance of completing the course corrections at the desired time, change as the distance decreases between the interceptor and the cruise missile, and between the cruise missile and its expected target**



- **This effect is very difficult to achieve by manipulating priorities**

The timeliness requirements for the interceptor missile control threads vary over the course of an engagement

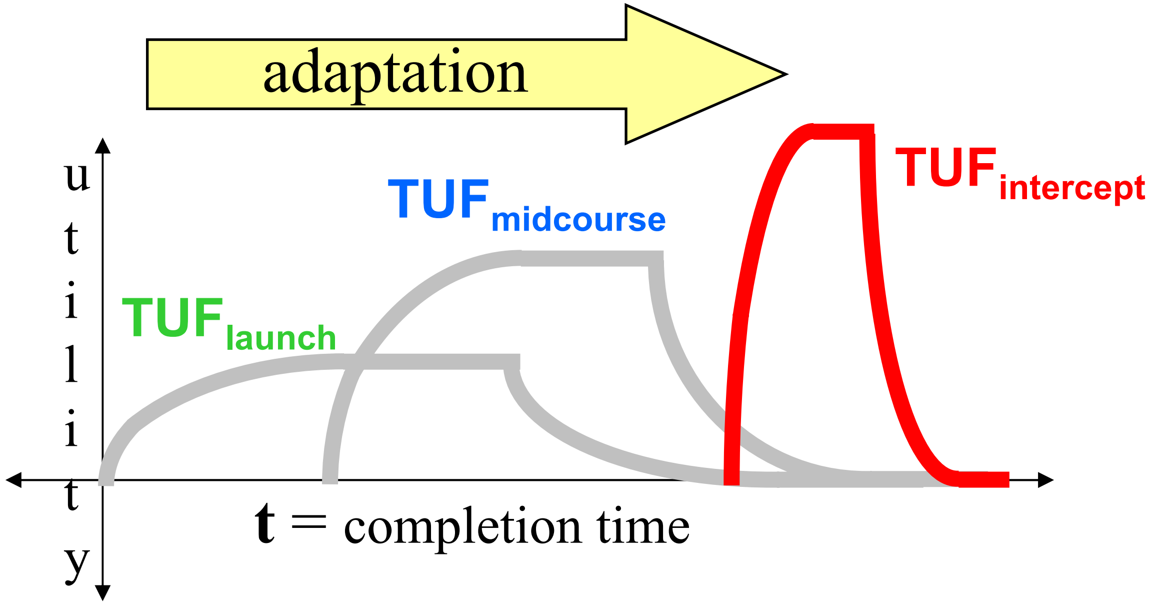
- After launch of the interceptor, the guidance control threads must issue timely repetitive course updates to ensure a successful intercept
- The required timeliness of these updates, and the importance of completing the course corrections at the desired time, change as the distance decreases between the interceptor and the cruise missile, and between the cruise missile and its expected target



- This effect is very difficult to achieve by manipulating priorities

The timeliness requirements for the interceptor missile control threads vary over the course of an engagement

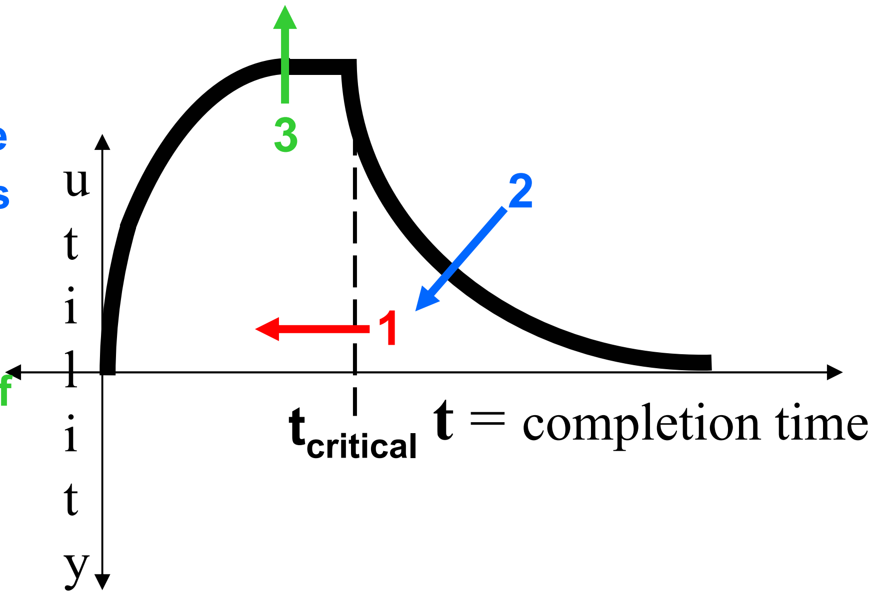
- After launch of the interceptor, the guidance control threads must issue timely repetitive course updates to ensure a successful intercept
- The required timeliness of these updates, and the importance of completing the course corrections at the desired time, change as the distance decreases between the interceptor and the cruise missile, and between the cruise missile and its expected target



- This effect is very difficult to achieve by manipulating priorities

The TUF's for the missile and interceptor control updates change dynamically during the mission

- 1. Variable critical (best) times – course corrections are needed more often as the distance between target and interceptor decreases
- 2. Variable "hardness" – it becomes more important to use the most recent position information as the distance between target and interceptor decreases
- 3. Dynamic maximum – the utility of successfully completing an intercept corresponds to the perceived threat of the target being intercepted



They also have variable importance depending on the threat potential of the target, independent of their timeliness

TUF/UA scheduling can be very cost/effective in adaptively achieving superior AQoS

- **TUF time constraints have been shown to be very natural, expressive, and powerful for the designers and programmers of the BM/C^2 applications we have experimented with**
- **But this paradigm does impose costs**
  - **TUF's are more complex than priorities in a sense**
    - **although they are derived directly from the application and are modular, they are functions instead of integers like priorities**
  - **UA scheduling is more complex than priority dispatching**
- **Various scheduling algorithms and application-specific engineering techniques can be used to trade off costs vs. effectiveness**
- **Continued research and application experimentation is needed**

A proof of concept software tool is being produced along with algorithms and formalism

- MITRE and our academic collaborator Virginia Tech are creating, simulating, implementing, measuring, and proving properties of, new UA algorithms for more cases
- We are also developing a proof of concept software tool for
  - creating and manipulating TUF's
  - plugging in various application-specific UA (and other) scheduling algorithms
  - simulating and analyzing the resulting schedules
- One version of this tool is being done in the context of an extant COTS real-time timing analysis product –

  the vendor is interested in the commercialization of our work
- Published and pre-published papers are available on my web site: www.real-time.org

Conclusion

- Application designers often think in terms of what we refer to as AQoS metrics, but not in a general and methodological way
- Instead, they consider certain metrics and use their domain expertise to attempt to aggregate these into the proper "tuning" of the system
- Thus, they've had few incentives to use their knowledge to understand and express behavioral options in the face of dynamic uncertainties (i.e., gracefully handling overloads) to facilitate automated resource management
- Time/utility functions are more natural, expressive, and realistic for dynamic systems, than priorities and deadlines
- AQoS metrics can be used to derive TUF's
- UA scheduling optimality criteria are powerful and adaptive
- TUF/UA based resource management has been shown to be very promising for dynamic systems such as BM/C^2