

Predictable End-to-End Timeliness in Network Centric Warfare Systems

E. Douglas Jensen

781-271-2514 • jensen@mitre.org



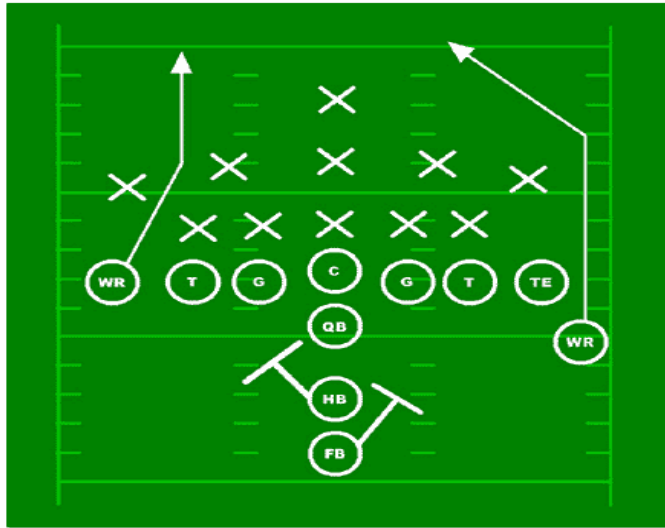
MITRE
Technology
Program

MITRE Sponsored Research

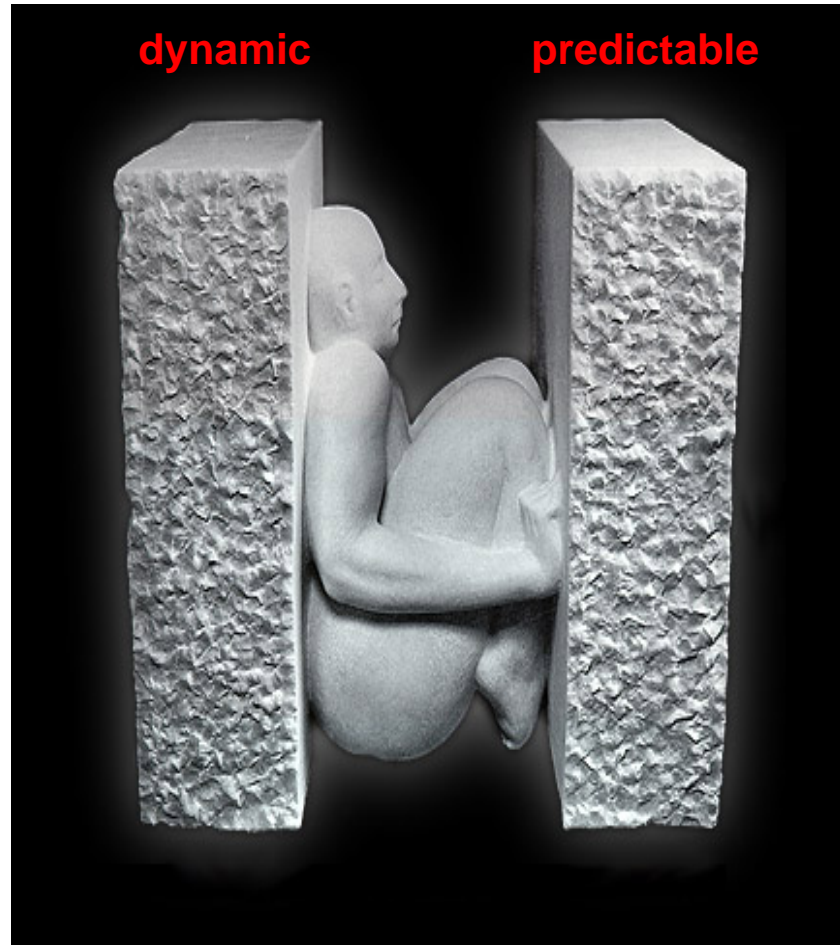
Problem: NCW systems are uncertain but must be predictable

- Commonly, many actions and information have *utility* that depends on their *timeliness*
- In distributed systems, action and information timeliness is often *end-to-end*
- Designers and users need to *reason dependably about (specify, manage, predict) timeliness* – critical in warfare
- Warfare systems and their environments are extremely *dynamic* with many *uncertainties* (mission, faults, loading, etc.)
- Reasoning about timeliness, especially end-to-end, is a very difficult unsolved problem in dynamic uncertain systems

Background: Football Is Like Combat



- Multi-level C²: owner, manager, coaches, quarterback
- Scripted plays
- Acceptably **predictable** end-to-end timeliness within a play
- But players are partially self-synchronized; plays and game are **dynamic** and emergent



MITRE

© 2007, The MITRE Corporation

Objective: Predictability Under Uncertainty

- **Enable designers and users of NCW systems and applications to**
 - **reason about predictability of end-to-end timeliness**
 - **receive formal assurances that timeliness and predictability will or will not be acceptable to them**

given the inherent dynamics and non-determinism due to overloads, failures, node mobility, varying network latencies and bandwidths, the exigencies of warfare, emergent behaviors, and the lack of overall system ownership and management

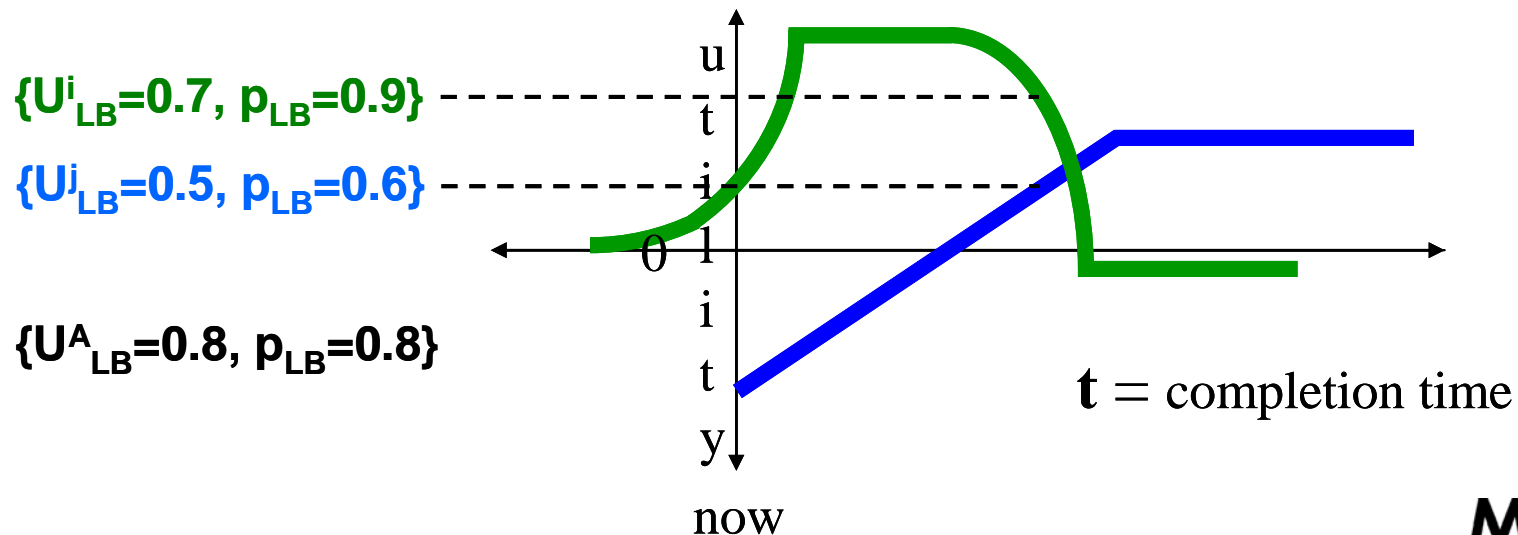
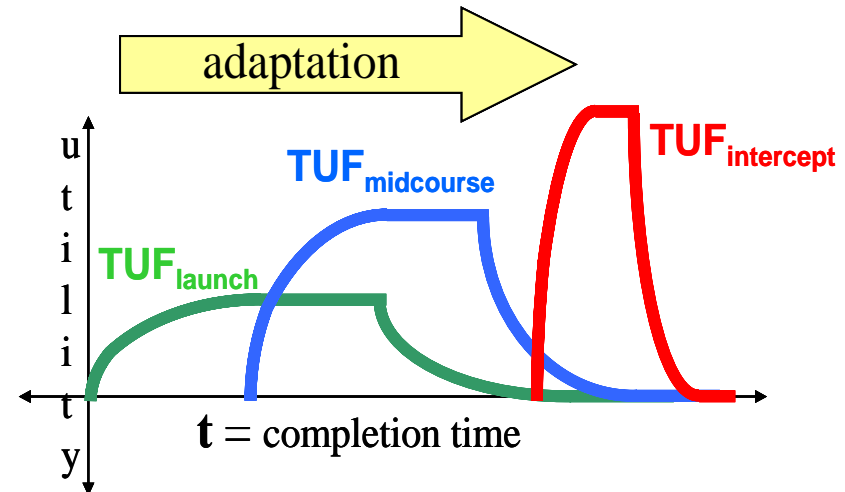
- **Technology transition to experimental DoD contexts**
- **MITRE technical stature (professional publications)**

Activities

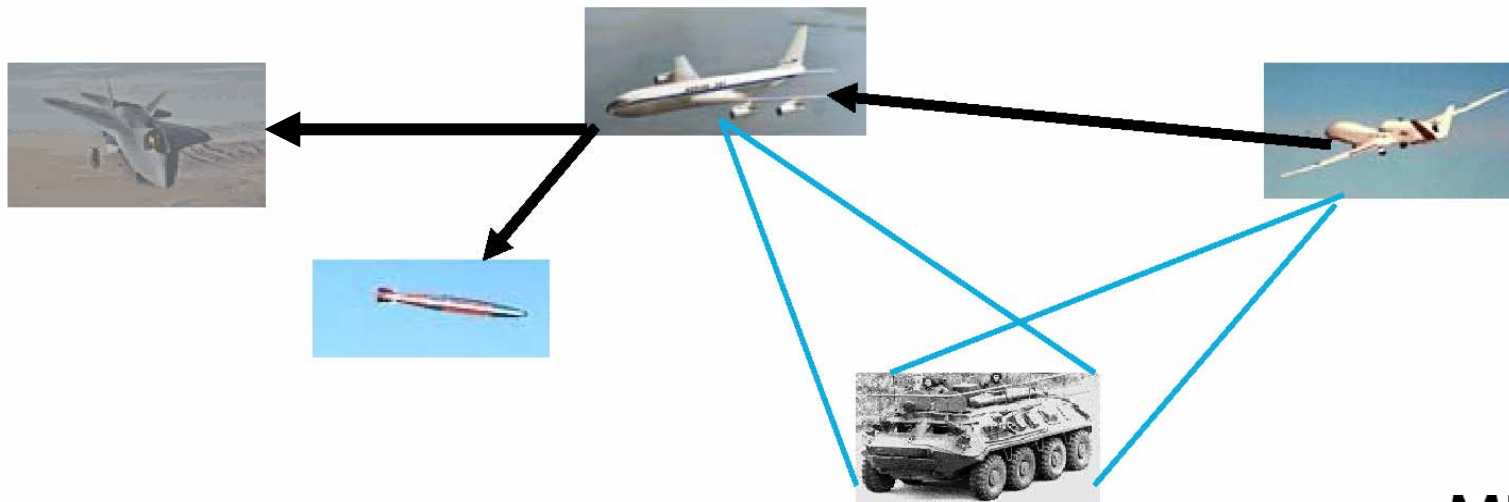
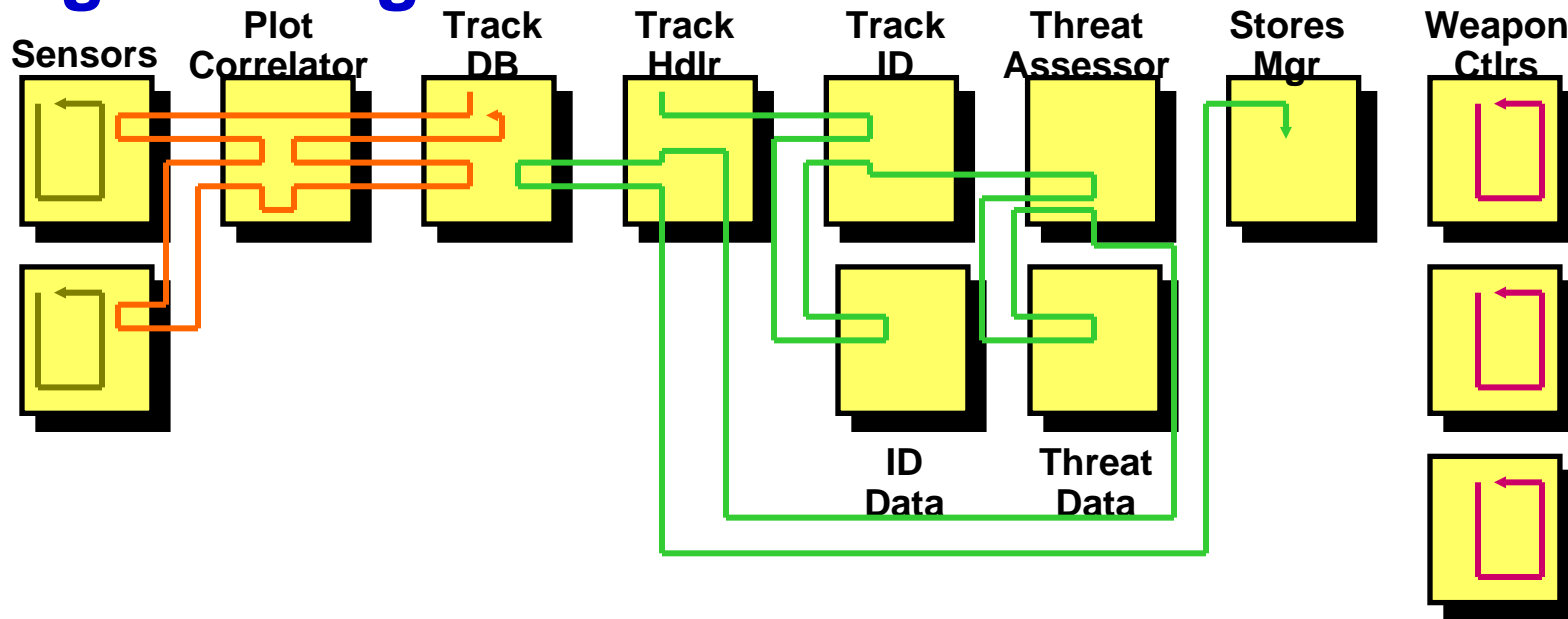
- **Devise, simulate, implement, and measure the performance of**
 - **distributed algorithms that manage resources**
 - **e.g., networks, sensors, processors, software**
 - **to formally assure non-deterministic predictability of timeliness under node and path failures in distributed systems**
- **Collaborate with, and co-advise, students at Virginia Tech**
 - **in FY06–07 thus far, six Ph.D. (three graduated), three M.S. (two graduated)**
- **Implement in Sun/MITRE's Distributed Real-Time Java (DRTSJ)**
 - **develop DRTSJ to the point that we need for our research**
- **Work with some C2C staff to choose credible NCW system, network, sensor, and application**
 - **operational characteristics**
 - **end-to-end timeliness requirements**

Highlight: Adaptive Probabilistic TUF Time Constraints

- Time/utility functions can be made to “shape shift” adaptively
- Time/utility functions can include specification of probabilistic lower bounds on
 - individual action/information utilities
 - accrued action/information utility



Highlight: Our Distributed Threads Programming Model



MITRE

© 2007, The MITRE Corporation

Impacts: on NCW, COTS, Research

- **Solutions to important open research problems for NCW**
 - Predictability of end-to-end timeliness under failures, using TUF/UA resource management with distributed threads, with formal assurance as strong as any particular dynamic context will allow
- **Enhancement, and first user, of Sun/MITRE's DRTSJ**
- **Technology demonstration and transition**
 - VA Tech's ONR-sponsored Advanced Wireless Integrated Navy Network
 - DoD program(s) are being explored
- **MITRE technical stature (conference and journal publications)**
 - 7 IEEE/ACM journal papers accepted thus far from 15 submitted; 13 IEEE/ACM conference papers accepted thus far from 19 submitted; 3 conference and 2 journal papers in preparation

Future Plans: Extend and Apply

- Concepts and algorithms
- Simulations
- Implementations
- Measurements
- Professional publications

Algorithm 5: insertByEDF()

```

1: input:  $T_i.PartialSched$ ,  $T_i.Dep$  and an ordered
   thread list  $\sigma_e$ ; output: the updated list  $\sigma_e$ ;
2: copy  $\sigma_e$  into  $\sigma_{tmp}$ ;  $\sigma_{tmp} := \sigma_e$ ;
3: Insert( $T_i$ ,  $\sigma_{tmp}$ ,  $T_i.X$ );
4:  $CuTT = T_i.X$ ;
5: for  $\forall T_j \in T_i.Dep$  from tail to head do
6:   if  $T_j \in \sigma_{tmp}$  then
7:      $TT = \text{lookup}(T_j, \sigma_{tmp})$ ;
8:     if  $TT < CuTT$  then break;
9:     else Remove( $T_j$ ,  $\sigma_{tmp}$ ,  $TT$ );
10:   $CuTT := \text{Min}(CuTT, T_j.X)$ ;
11:  Insert( $T_j$ ,  $\sigma_{tmp}$ ,  $CuTT$ );
12: if feasible( $\sigma_{tmp}$ ) then
13:    $\sigma_e := \sigma_{tmp}$ ;
14: return  $\sigma_e$ ;

```

