Overview of a system component

Distributed components communicate via input and output streams
Inside the box…

Target Platform
1. Streaming application tasks

![Diagram of streaming application tasks]

- DCT: Discrete Cosine Transform and Quantisation
- IQDCT: Inverse Quantisation Discrete Cosine Transform

2. Heterogeneous computing and memory resources

![Diagram of heterogeneous computing resources]

- Image Coprocessor
- DSP
- RISC
- FPGA
- CAN Interface
3. Heterogeneous RTOS scheduling and synchronization protocols

- TDMA
- FCFS
- EDF
- Proportional Share
- Dynamic Fixed Priority
- Static

4. Heterogeneous communication resources

- Topology (ring, mesh, star)
- Switching strategies (packet, circuit)
- Routing strategies (static, dynamic, reconfigurable)
- Arbitration policies (dynamic, TDM, CDMA)
The Design Problem

Build a system from subsystems that satisfy the application’s requirements and resource constraints.
The Performance Analysis Problem

- Compute/verify the performance properties of the system model
  - Maximum fill-level (backlog) of the buffers?
  - Maximum end-to-end delay of the stream?
  - Characteristics of the output stream?
  - Characteristics of the remaining resource?
Key Challenges: Complex Event Streams

- Infinite sequence of items (events)
- Highly bursty
- Events of multiple types interleaving
- Varied memory and execution demand
- ...

Key Challenges: Complex Tasks & Architectures

- Complex processing semantics
  - fill-level of the buffers
  - synchronization between different streams

- Heterogeneous computing and communication resources

- Various scheduling policies
  - EDF, Fixed-Priority, TDMA, etc.
  - complex state-dependent scheduling schemes
Complex Trade-offs

- Computational Demand
- Throughput
- Memory Size

Two Categories of Performance Analysis

- Simulation:
  - Input trace
  - Output trace
  - Analysis bound

- Formal Analysis:
  - Abstract input model
  - Analysis bound

Hybrid of Simulation & Formal Analysis
Simulation vs Formal Analysis

Our focus!

Formal Analysis

Real System

Simulation

Formal Analysis Overview

Concrete System

Input Streams

A Processing Component

Output Streams
Formal Analysis Overview

How event streams arrive and its characteristics

Formal Analysis Overview

How much and when the resource are available
Formal Analysis Overview

Formal Models and Analysis Methods

1 Standard Event Models (SEM)
   - periodic, periodic with jitter/burst
   - based on classical scheduling theory

👍 Simple, easy to analyze

👎 Too restrictive

👎 Pessimistic results
**Standard Event Models**

- Input data items arrive regularly, one data item (event) every $P$ time units.

**Distributed Real-Time Systems**

- Sensed data items (events) arrive irregularly, may come in burst.
- Pessimistic analysis results if assuming classical models.

- Periodic task model: ?
2 Queing theory and variations
   - e.g., Real-Time Calculus (RTC)
   - streams & resources: functions
   - analysis: (min,+ ) and (max,+ ) algebra

   🍀 Capture burstiness of streams & resource availability
   🍀 Highly efficient
   🚫 Cannot model state-dependencies

3 Automata-based models
   - Timed automata, event count automata, dataflow graphs, etc.

   🍀 Models state-dependencies
   🍀 Highly accurate
   🚫 Large systems ➔ inefficient
4 Hybrid Models and Methods
- RTC + SEM
- RTC + ECA
- Multi-Mode RTC
- ...

👍 Good accuracy-efficiency trade-off

The rest of the talk...

Formal Analysis using Real-Time Calculus (RTC)
RTC Background

- Originated from Network Calculus in computer networks domain
  - extended for real-time embedded systems
- Worst-case deterministic formal analysis
  - variant of classical queuing theory
- Abstract models: count-based abstraction
- Analysis: min-plus / max-plus algebra

Recall…

[Diagram showing a processing component and related models]
RTC Performance Model

An Arrival Pattern

$R(t) = \text{number of events that arrive in } [0, t)$

$\text{# events that arrive in } [t, t+\Delta) \text{ is: } R(t+\Delta) - R(t)$
## Count-based Abstraction

A set of arrival patterns

<table>
<thead>
<tr>
<th>$\Delta$</th>
<th>Lower bound</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\ddots$</td>
<td>$\ddots$</td>
</tr>
</tbody>
</table>

A concrete time instant

## Load Model: Arrival Functions

$\alpha = (\alpha^l, \alpha^u)$

An arrival pattern $R(t)$ satisfies $\alpha$ iff

$$\alpha^l(\Delta) \leq R(t+\Delta) - R(t) \leq \alpha^u(\Delta)$$
A Service Pattern

\[ C(t) = \text{number of events that can be processed in } [0,t) \]

\[ C(t) = C(t+\Delta) - C(t) \]

Service Model: Service Functions

\[ \beta = (\beta^l, \beta^u) \]

<table>
<thead>
<tr>
<th>( \Delta )</th>
<th>( \beta^l(\Delta) )</th>
<th>( \beta^u(\Delta) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
</tbody>
</table>

A service pattern \( C(t) \) satisfies \( \beta \) iff

\[ \beta^l(\Delta) \leq C(t+\Delta) - C(t) \leq \beta^u(\Delta) \]
Units of Arrival and Service Functions

- \([R(t), \alpha(\Delta)]\) and \([C(t), \beta(\Delta)]\) can also be specified in terms of the number of resource units
  - processor cycles, transmitting bit, etc.

- Should *always* convert to the same unit before performing analysis

Examples of Arrival and Service Functions
Periodic Event Streams

- Periodic
  - p

- Periodic with burst
  - p
  - j
  - ≥ d

\[ p: \text{period} \quad j: \text{jitter} \quad d: \text{minimum inter-arrival distance} \]

Arrival Function of Periodic Event Streams

\[ \alpha^l(\Delta) = \left\lfloor \frac{\Delta}{p} \right\rfloor \quad \alpha^u(\Delta) = \left\lceil \frac{\Delta}{p} \right\rceil \]

# events

\[ \Delta \]

Periodic arrival function

\[ \alpha^l \quad \alpha^u \]

\[ 0 \quad p \quad 2p \quad 3p \quad 4p \quad 5p \]
P = 3: exactly 1 event arrives every 3 time units

→ For any given t, exactly 1 event arrives in the interval [t, t+3)

\[ \alpha^u(2) = 1, \; \alpha^l(2) = 0 \]
\[ \alpha^u(3) = 1, \; \alpha^l(3) = 1 \]

**Periodic with Jitter Event Streams**

Suppose min distance between any two events is 0, then

In any interval of length \( \Delta \): at most \( \left\lfloor \frac{\Delta + j}{p} \right\rfloor \) events

at least \( \left\lceil \frac{\Delta - j}{p} \right\rceil \) events
$P = 3, J = 2$: at most 2, at least 0 in $[t, t+3)$

$\alpha^u(3) = \lceil (3 + 2)/3 \rceil = 2$

$\alpha^l(3) = \lfloor (3 - 2)/3 \rfloor = 0$

$\alpha^{l}(\Delta) = \left\lfloor \frac{\Delta - j}{p} \right\rfloor$

$\alpha^{u}(\Delta) = \min\{ \left\lceil \frac{\Delta + j}{p} \right\rceil , \left\lfloor \frac{\Delta}{d} \right\rfloor \}$
Common Resources

- **full resource service function**
  - \( \beta^u = \beta^l \)
  - The processor is always available
  - \( f = \) processor frequency

- **bounded delay service function**
  - \( fD \)
  - \( \beta^u \) and \( \beta^l \)
**TDMA Resource**

- A shared resource of bandwidth $B$
- $n$ applications: $App_1$, $..., App_n$
- TDMA policy
  - a resource slot of length $s_i$ is assigned to $App_i$ in every cycle of length $c$
  - the resource given to $App_i$ is bounded by

\[
\beta_i^l(\Delta) = B \max \left\{ \frac{\Delta}{c} s_i, \Delta - \left\lfloor \frac{\Delta}{c} \right\rfloor (c - s_i) \right\}
\]
\[
\beta_i^u(\Delta) = B \min \left\{ \left\lceil \frac{\Delta}{c} \right\rceil s_i, \Delta - \left\lceil \frac{\Delta}{c} \right\rceil (c - s_i) \right\}
\]
Previous Lecture…

- General concepts of the design and performance analysis of distributed real-time embedded systems
- Simulation vs formal analysis
- Existing formal analysis methods: pros and cons
- **Real-Time Calculus (RTC)**
  - High-level overview
  - Count-based abstraction
  - Definition of arrival and service functions
Real-Time Calculus (cont.)

• A brief introduction to RTC
  – Refer to reading list for more!

• Materials are based on
  – Le Boudec and Thiran’s book on Network Calculus
  – The MPA framework

Recall... Event Streams

• Infinite sequences of data items (events)

• A concrete arrival pattern can be described as a cumulative function \( R(t) \)
  – \( R(t) = \) number of items arrive in the time interval \([0,t)\)

• All possible arrival patterns of an event stream is abstracted as an arrival function \( \alpha(\Delta) \)
Arrival Function of A Set of Concrete Patterns

Recall... Resources

- A concrete service pattern
  - how much and when the resource is available
  - captured as a cumulative function $C(t)$ which gives the amount of resource units available in time interval $[0,t)$

- All possible service patterns of a resource is abstracted as a service function $\beta(\Delta)$
Service Function of A Set of Concrete Patterns

RTC Performance Model

backlog = \textit{Buf}(\alpha, \beta)
delay = \textit{Del}(\alpha, \beta)

The functions \( f_\alpha, f_\beta, \text{ Buf, Del} \) must take into account the scheduling policy and the processing semantics of the component.
Processing Model: Abstract Component

• Relate input arrival/service functions and
  – output arrival and service functions
  – maximum backlog
  – maximum delay

• The computation must capture the way input event streams are processed by the resource

• Vary depending on the scheduling policy and processing semantics, but always deterministic

A concrete system component

\[ R(t) \rightarrow GPC \rightarrow C(t) \rightarrow R'(t) \rightarrow C'(t) \]

an arrival pattern of the input stream
a service pattern of the available resource
a service pattern of the remaining resource
an arrival pattern of the output stream
Greedy Processing Component

- Triggered by incoming events
- Events are processed in a greedy fashion and FIFO order
  - subjected to resource availability
  - waiting events are stored in the input buffer
- Backlog at time $t$
  - $B(t) = $ #events in the buffer at time $t$
- Delay at time $t$
  - $d(t) = $ the maximum processing time (including waiting time) of an event arriving before $t$

![Diagram of Greedy Processing Component](image.png)

- $R(t)$
- $C(t)$
- $C'(t)$
- $G$PC
- $R'(t)$

- continue to process again
- no cycle to process
- unused resource
- no additional output events
- buffer empty here!
GPC: Output Stream

\[ R'(t) = \inf_{0 \leq u \leq t} \{ R(u) + C(t) - C(u) \} \]

For all \( u \leq t \):
- \( R'(u) \leq R(u) \) and \( R'(t) \leq R'(u) + C(t) - C(u) \)
  - #output-events in \([0,u)\) is no more than #input-events in \([0,u)\)
  - #output-events in \([u,t)\) is no more than #events that can be processed in \([u,t)\)

Hence, \( R'(t) \leq R(u) + C(t) - C(u) \)

- Let \( u_0 \) be the last instant before \( t \) at which \( B(u_0) = 0 \)
  - \( R'(u_0) = R(u_0) \); \( R'(t) = R(u_0) + C(t) - C(u_0) \)
  - Thus, \( R'(t) = R'(u_0) + C(t) - C(u_0) \)

Conservative use of resource:
\[ C(t) = C'(t) + R'(t) \]

GPC: Remaining Resource
Backlog at time $t$: $B(t) = R(t) - R'(t)$

Maximum backlog: $B_{\text{max}} = \max_{t \geq 0} B(t)$

Delay at time $t$:
$$d(t) = \min\{\lambda : R'(t+\lambda) \geq R(t)\}$$
An abstract system component

- $\alpha(\Delta)$: Arriving function of the input stream
- $\beta(\Delta)$: Service function of the available resource
- $\beta'(\Delta)$: Service pattern of the remaining resource
- $\alpha'(\Delta)$: Arriving function of the output stream

Basic Min-plus/Max-plus Operators

- **Min-plus convolution and de-convolution**
  
  $$ (f \otimes g)(t) = \inf_{0 \leq u \leq t} \{ f(t - u) + g(u) \} $$

  $$ (f \odot g)(t) = \sup_{u \geq 0} \{ f(t + u) - g(u) \} $$

- **Max-plus convolution and de-convolution**
  
  $$ (f \bar{\otimes} g)(t) = \sup_{0 \leq u \leq t} \{ f(t - u) + g(u) \} $$

  $$ (f \bar{\odot} g)(t) = \inf_{u \geq 0} \{ f(t + u) - g(u) \} $$
GPC: Output Bounds

\[ \alpha'^u = \min \{ (\alpha^l \otimes \beta^u) \otimes \beta^l, \beta^u \} \]

\[ \alpha'^u = \min \{ (\alpha^u \otimes \beta^u) \otimes \beta^l, \beta^u \} \]

\[ \beta'^u = (\beta^l - \alpha^u) \otimes 0 \]

\[ \beta'^u = (\beta^u - \alpha^l) \otimes 0 \]

**Compute \( \alpha'^u \) - Intuitive Idea**

\[ \text{max\_output}(\Delta+\lambda) \leq \text{max\_input}(\Delta+\lambda-\tau) + \text{max\_processed}(\tau), \forall 0 \leq \tau \leq \Delta+\lambda \]

\[ \leq \sup_{0 \leq \tau \leq \Delta+\lambda} \{ \alpha'^u(\Delta+\lambda-\tau) + \beta'^u(\tau) \} = \gamma(\Delta+\lambda), \text{with } \gamma = \alpha'^u \otimes \beta'^u \]

\[ \text{max\_output}(\Delta) \leq \text{max\_output}(\Delta+\lambda) - \text{min\_processed}(\lambda), \forall \lambda \geq 0 \]

\[ \leq \gamma(\Delta+\lambda) - \beta'^l(\lambda), \forall \lambda \geq 0 \]

\[ \leq \inf_{\lambda \geq 0} \{ \gamma(\Delta+\lambda) - \beta'^l(\lambda) \} = (\gamma \otimes \beta'^l)(\Delta) \]

\[ \leq (\alpha'^u \otimes \beta'^u) \otimes \beta'^l(\Delta) \]

Further, \( \text{max\_output}(\Delta) \leq \text{max\_processed}(\Delta) \leq \beta'^u(\Delta) \)

\[ \Rightarrow \alpha'^u \leq \min \{ (\alpha^u \otimes \beta^u) \otimes \beta^l, \beta^u \} \]
GPC: Backlog and Delay Bounds

\[ B_{\text{max}} = \sup_{t \geq 0} \{ R(t) - R'(t) \} \leq \sup_{\Delta \geq 0} \{ \alpha^u(\Delta) - \beta^l(\Delta) \} \]

\[ D_{\text{max}} = \sup_{t \geq 0} \{ \inf \{ u \geq 0 : R(t) \leq R'(t + u) \} \} = \sup_{\Delta \geq 0} \{ \inf \{ u \geq 0 : \alpha^u(\Delta) \leq \beta^l(\Delta + u) \} \} \]
Scheduling Multiple Event Streams

- The video stream has higher priority than the audio stream.
  - Process the video stream first.
- Remaining resources are used to process the audio stream.

Fixed Priority:

- Video stream has higher priority than audio stream.
  - Process the video stream first.
- Remaining resources are used to process the audio stream.
TDMA Scheduling

\[ \beta_i : \text{computed based on the length of the TDMA cycle } c \text{ and the slot } S_i \]

\[
\beta^i_1(\Delta) = B \max \left\{ \left\lfloor \frac{\Delta}{c} s_i \right\rfloor, \Delta - \left\lfloor \frac{\Delta}{c} (c - s_i) \right\rfloor \right\}
\]

\[
\beta^i_2(\Delta) = B \min \left\{ \left\lfloor \frac{\Delta}{c} s_i \right\rfloor, \Delta - \left\lfloor \frac{\Delta}{c} (c - s_i) \right\rfloor \right\}
\]

Modular Performance Analysis using RTC
Mixed Hierarchical Scheduling

The RTC Toolbox

www.mpa.ethz.ch/rtctoolbox
RTC - Summary

- **Modeling: count-based abstraction**
  - captures burstiness of event streams and variability of the resources as functions

- **Analysis: min-plus and max-plus algebra**
  - can be computed efficiently with tool support

- **Modular and compositional**
  - possible combination with other methods, e.g. standard event models, ECA, simulation

- **Modeling of state-dependencies is difficult**
  - extension of RTC: an active area of study
  - various work combines concepts in RTC with automata

References and Readings

Real-Time Calculus:

1. Jean-Yves Le Boudec and Patrick Thiran: "Network Calculus", Lecture Notes in Computer Science 2050, Springer Verlag, January 2004 (Chapter 1 & 3)

References and Readings

Other Formal Analysis Methods

1. Kai Richter and Rolf Ernst: "Event Model Interfaces for Heterogeneous System Analysis", Design Automation and Test in Europe Conference (DATE), 2002


4. C. Norstrom, A. Wall and W. Yi: "Timed automata as task models for event-driven systems", 6th International Workshop on Real-Time Computing and Applications Symposium (RTCSA), Hong Kong, China, 1999

Hybrids of RTC and others:


