جَـامعة
المَـنارة
MANARA UNIVERSITY

محاضرة3 د.فادي متوج

DATAFLOW MODELS

1. Dataflow Models: an Example
2. Kahn Process Networks: a Deterministic Model
3. Synchronous Dataflow: Statically Schedulable Dataflow Models
4. Deriving a static Schedule for Synchronous Dataflow Models

Dataflow Models

- Systems are specified as directed graphs where:
  - *nodes* represent computations (processes);
  - *arcs* represent totally ordered sequences (streams) of data (tokens).
- Depending on their particular semantics, several models of computation based on dataflow have been defined:
  - Kahn process networks
  - Dataflow process networks
  - Synchronous dataflow
  - - - - - - -
- Dataflow models are suitable for signal-processing algorithms:
  - Code/decode, filter, compression, etc.
  - Streams of periodic and regular data samples

ما هي نماذج تدفق البيانات؟

يمكن التفكير فيها كأنها وصف لطريقة عمل نظام ما، ولكن التركيز هنا ليس على تسلسل الأوامر (مثل معظم البرامج التقليدية التي نكتبها)، بل على **حركة البيانات** نفسها.

يتم تمثيل النظام **كرسم بياني موجّه (Directed Graph)** هذا يعني أنه لدينا:

1. **عُقد (Nodes) :** تمثل **العمليات الحسابية (Computations)** أو المهام التي يجب إنجازها. يمكننا تخيلها كعمال في خط تجميع، كل عامل متخصص في مهمة معينة (تركيب قطعة، فحص جودة، تغليف...). في عالم الأنظمة المدمجة، هذه قد تكون دالة برمجية، عملية مستقلة(process) ، أو حتى وحدة هاردوير متخصصة.
2. **أقواس أو مسارات (Arcs) :**تمثل **تدفق البيانات (Streams)** بين العمليات. هي مثل السيور الناقلة في المصنع التي تنقل القطع بين العمال. البيانات هنا تسمى أحياناً **"Tokens"** و هي **"totally ordered sequences"**، أي أن البيانات تصل بنفس الترتيب الذي أُرسلت به، تماماً كما أن القطع على السير الناقل تحافظ على ترتيبها.

لماذا هذا النموذج مهم؟

لأنه يعكس طبيعة الكثير من الأنظمة، خاصة تلك التي تتعامل مع **معالجة الإشارات (Signal Processing)** . لنأخذ مثلا نظاماً يقوم بمعالجة الصوت القادم من ميكروفون في روبوت:

- بيانات الصوت الخام تتدفق من الميكروفون (هذاstream)
- تصل إلى عقدة (Node) تقوم بتصفية الضجيج (Computation) .
- البيانات المصفاة (stream جديد) تذهب إلى عقدة أخرى تقوم بتحليل الترددات للتعرف على أمر صوتي (Computation)
- وهكذا...

الجميل في هذا النموذج أن **العملية (Node) تبدأ عملها فقط عندما تتوفر لديها البيانات اللازمة على مداخلها (Arcs)**. لا يوجد تحكم مركزي يقول "ابدأ الآن"، بل تدفق البيانات هو الذي يحرك النظام.

أنواع مختلفة من نماذج تدفق البيانات:

- Kahn process networks
- Dataflow process networks
- Synchronous dataflow (SDF)

هذه ليست مجرد أسماء مختلفة لنفس الشيء، بل لكل منها **قواعد دلالية (Semantics)** خاصة بها. أي أن لكل نموذج قواعده التي تحدد بالضبط:

- متى يمكن للعقدة أن "تستهلك" البيانات من مدخلاتها؟
- كم عدد البيانات التي تستهلكها وتنتجها في كل مرة؟
- هل هناك قيود على حجم الـ "buffers" بين العقد؟
- كيف يتم التعامل مع التوقيت والتزامن؟ (وهذا مهم جداً في Synchronous Dataflow الذي سنتحدث عنه لاحقاً).

لماذا هي مناسبة لمعالجة الإشارات؟

كما ذكرنا، تطبيقات مثل:

- التشفير وفك التشفير(Code/decode)
- الفلاتر(Filter)
- الضغط (Compression)

كلها تتضمن أخذ **تدفقات من العينات المنتظمة والدورية - (Streams of periodic and regular data samples)**

مثل عينات الصوت الرقمية، بكسلات الصورة، قراءات الحساسات المتتالية - وتطبيق سلسلة من العمليات عليها.

نموذج تدفق البيانات يوفر طريقة طبيعية ومنظمة لوصف هذه العمليات المتسلسلة.

في مجال الروبوتات مثلا، يتم التعامل بكثافة مع بيانات قادمة من حساسات متعددة (كاميرات، ليدار، IMU، ميكروفونات...) هذه البيانات تأتي كتدفقات (streams) حيث سنختاج لتصميم أنظمة تقوم بـ:

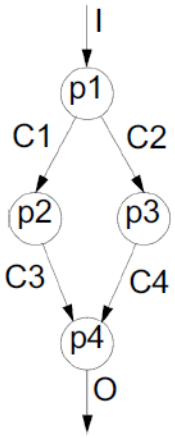
- **تصفية (Filtering)** :إزالة الضجيج من قراءات الحساسات.
- **دمج (Fusion)** :دمج بيانات من حساسات مختلفة للحصول على فهم أفضل للبيئة.
- **استخلاص الميزات (Feature Extraction)** :مثلاً، استخلاص حواف أو زوايا من صور الكاميرا.
- **التعرف على الأنماط (Pattern Recognition)** :مثل التعرف على الأوامر الصوتية أو الأجسام في الصورة.

كل هذه المهام يمكن نمذجتها وتصميمها بكفاءة باستخدام نماذج تدفق البيانات. فهي تساعد على تنظيم المعالجة المعقدة للبيانات وتسهيل تصميم أنظمة متوازية (parallel) وموزعة(distributed) .

نماذج تدفق البيانات هي أداة قوية لوصف وتصميم الأنظمة التي تعالج تدفقات البيانات، حيث تمثل العمليات كعقد والبيانات المتدفقة كأقواس تربط بينها. وهي مناسبة بشكل خاص لتطبيقات معالجة الإشارات الموجودة بكثرة في الروبوتات .

Dataflow Models

```
Process p1( in int a, out int x, out int y) {
.....
}
Process p2( in int a, out int x) {
.....
}
Process p3( in int a, out int x) {
.....
}
Process p4( in int a, in int b, out int x) {
.....
}
channel int I, O, C1, C2, C3, C4;
p1(I, C1, C2);
p2(C1, C3);
p3(C2, C4);
p4(C3, C4, O);
```



- **The internal computation of a process can be specified in any programming language (e.g. C).**

  This is called the *host language*.

نماذج تدفق البيانات: هي طريقة معينة لنمذجة أو وصف سلوك النظام المدمج، تركز على كيفية تدفق البيانات بين أجزاء النظام المختلفة. لنتخيل نظام روبوت ما، البيانات تتدفق من الكاميرا أو حساس المسافة، تمر بمعالجة، ثم تذهب لتحريك المحركات. هذا مثال بسيط لتدفق بيانات.

وصف العمليات (Processes): على اليسار، نرى تعريفاً لأربع عمليات p1, p2, p3, p4 :

- كل عملية لها مدخلات (inputs) ومخرجات (outputs)
  - in int a : تعني أن العملية تستقبل بيانات من النوع int (عدد صحيح) كمدخل اسمه a
  - out int x, out int y : تعني أن العملية تنتج بيانات من النوع int كمخرجات بأسمائها x و y
- النقاط {...} داخل كل عملية تشير إلى **المنطق الداخلي** لهذه العملية. هذا المنطق هو الذي يحدد ماذا تفعل العملية بالبيانات المدخلة لتنتج البيانات المخرجة. قد يكون هذا المنطق عبارة عن معادلات رياضية، عمليات منطقية، أو أي حسابات أخرى.

في مجال الروبوتات : بفرض p1 هي عملية قراءة بيانات من مستشعر مسافة، p2 هي عملية حساب المسافة الآمنة بناءً على القراءة، p3 هي عملية تحديد زاوية الدوران المطلوبة، و p4 هي عملية إصدار أوامر للمحركات بناءً على المسافة والزاوية. كل عملية تأخذ بيانات وتخرج بيانات أخرى معالجة.

تعريف القنوات (Channels) :

- القنوات هي **الأنابيب** أو **الممرات** التي تتدفق البيانات عبرها بين العمليات. يمكن تخيلها كشرايين النظام.

- int : نوع البيانات التي ستتدفق عبر هذه القنوات (في هذه الحالة، أعداد صحيحة).
- I, O, C1, C2, C3, C4: هذه هي أسماء القنوات .
  - I : قناة المدخلات الرئيسية للنظام ككل.
  - O : قناة المخرجات الرئيسية للنظام ككل.
  - C1, C2, C3, C4 : قنوات داخلية تربط بين العمليات المختلفة.

ربط العمليات بالقنوات (الشبكة) :الأسطر التالية توضح كيف تتصل العمليات ببعضها البعض عبر القنوات، مشكلةً شبكة تدفق البيانات:

- p1(I, C1, C2); : العملية p1 تأخذ مدخلها من القناة I، وتخرج مخرجاتها إلى القناتين C1 و C2 (نلاحظ ترتيب الأسماء يطابق تعريف العملية: in a يقابل I، out x يقابل C1، out y يقابل C2).
- p2(C1, C3); العملية p2 تأخذ مدخلها من القناة C1، وتخرج مخرجاتها إلى القناة C3 .
- p3(C2, C4); : العملية p3تأخذ مدخلها من القناة C2، وتخرج مخرجاتها إلى القناة C4 .
- p4(C3, C4, O); :العملية p4تأخذ مدخلاتها من القناتين C3وC4، وتخرج مخرجاتها إلى القناة O .

الرسم البياني هو تمثيل بصري ممتاز لشبكة تدفق البيانات هذه.

- الدوائر تمثل **العمليات** (p1, p2, p3, p4)
- الأسهم تمثل **القنوات** واتجاه تدفق البيانات .
  - السهم الداخل إلى p1 والمسمى I يمثل قناة الإدخال الرئيسية.
  - السهم الخارج من p4 والمسمى O يمثل قناة الإخراج الرئيسية.
  - الأسهم بين الدوائر تمثل القنوات الداخلية(C1, C2, C3, C4)
- **"The internal computation of a process can be specified in any programming language (e.g. C). This is called the host language."**
  - نموذج تدفق البيانات يصف كيف تتصل العمليات وتتبادل البيانات، لكنه لا يفرض علينا لغة برمجة معينة لكتابة المنطق الداخلي لكل عملية. يمكننا استخدام لغةC ، C++، بايثون، أو أي لغة مناسبة لكتابة الكود الفعلي الذي تقوم به كل عملية (مثل قراءة البيانات، إجراء الحسابات، اتخاذ القرارات). هذه اللغة التي تكتب بها المنطق الداخلي تسمى "لغة الاستضافة (host language) ".
  - **في مجال الروبوتات :**هذا يعني أننا يمكن أن نصمم الهيكل العام لنظام الروبوت الخاص بنا باستخدام نموذج تدفق البيانات، ثم نقوم بكتابة كود معالجة الصور بلغة سواء C++ أو Python حسب المكتبات المتاحة، وكود التحكم في المحركات بلغة C مثلاً على المتحكم الصغري. النموذج يجمع كل هذا معاً.

إن نماذج تدفق البيانات هي طريقة قوية لتقسيم نظامنا المدمج إلى أجزاء أصغر (عمليات) تتواصل مع بعضها البعض عن طريق تبادل البيانات عبر قنوات. هذا يساعد في:

- **فهم النظام** :يسهل رؤية مسار تدفق البيانات.
- يمكننا تطوير كل عملية بشكل مستقل.
- **التوازي** :في كثير من الأحيان، يمكن تنفيذ هذه العمليات بالتوازي إذا كانت الأجهزة تدعم ذلك، مما يحسن الأداء (وهذا مهم جداً في الروبوتات التي تحتاج استجابة سريعة).

Kahn Process Networks (KPN)

- **Processes communicate by passing data tokens through unidirectional FIFO channels.**
- **Writes to the channel are non-blocking.**
- **Reads are blocking:**
  - **the process is blocked until there is sufficient data in the channel**

A process that tries to read from an empty channel waits until data is available. It cannot ask whether data is available *before* reading and, for example, if there is no data, decide not to read that channel. → **DETERMINISM**

شبكات عمليات كان(Kahn Process Networks - KPN) : هو نموذج سمي على اسم العالم David Kahn و هو أحد أشهر نماذج تدفق البيانات المستخدمة في الأنظمة المدمجة والأنظمة المتوازية.

بعد أن فهمنا المفهوم العام لتدفق البيانات والعمليات والقنوات، سنحدد القواعد الدقيقة التي تحكم كيفية تفاعل العمليات وتبادل البيانات في نموذج KPN وهذا مهم جداً لنفهم سلوك النظام بشكل صحيح..

: "Processes communicate by passing data tokens through unidirectional FIFO channels."

- **Processes communicate** : العمليات تتواصل مع بعضها البعض.
- **passing data tokens** : تتواصل عن طريق تمرير "رموز بيانات (data tokens) ". يمكن تخيل التوكن كحزمة صغيرة من البيانات (مثل قراءة حساس واحد، أو قيمة محسوبة واحدة). هي الوحدة الأساسية للبيانات التي تتدفق.
- **unidirectional** : أحادية الاتجاه. القنوات تسمح للبيانات بالتدفق في اتجاه واحد فقط، من عملية مرسلة إلى عملية مستقبلة. لا يمكن للبيانات أن تعود في نفس القناة.

- **FIFO channels :** قنوات تعمل بنظام "من يدخل أولاً يخرج أولاً (First-In, First-Out) " .هذا يعني أن التوكنات التي تُرسل إلى القناة تخرج منها بنفس الترتيب الذي دخلت به. لا يمكن لتوكن أُرسل لاحقاً أن يتجاوز توكن أُرسل قبله في نفس القناة. هذه الخاصية حاسمة جداً لضمان السلوك المتوقع للنظام.

في مجال الروبوتات :بفرض أنه يتم إرسال تسلسل زمني لقراءات حساس إلى عملية معالجة. نظام FIFO يضمن أن قراءة الوقت t1 ستعالج قبل قراءة الوقت t2 إذا تم إرسالهما بهذا الترتيب، وهذا ضروري للحفاظ على تزامن البيانات وصحة المعالجة.

- **Writes are non-blocking :**هذا يعني أن العملية التي تريد إرسال (كتابة) توكن إلى قناة ما **لا تتوقف ولا تنتظر** حتى يتم استقبال التوكن من قبل العملية الأخرى. بمجرد أن تقرر العملية الإرسال، فإنها تضع التوكن في القناة بافتراض أن هناك سعة في القناة، رغم أن نموذج KPN النظري يفترض قنوات بسعة لا نهائية لتبسيط التحليل وتستمر في عملها مباشرة. هي لا تهتم إذا كانت العملية المستقبلة جاهزة للقراءة أم لا.
- **Reads are blocking :**على النقيض تماماً من الكتابة، عملية **القراءة** من القناة هي عملية **مانعة (blocking)** .
- **"the process is blocked until there is sufficient data in the channel"** : العملية التي تحاول قراءة توكن من قناة **تتوقف وتنتظر** حتى يصبح التوكن متوفراً في القناة. إذا كانت القناة فارغة، فإن العملية القارئة "تنحظر (blocks) "ولا تستطيع إكمال عملها حتى تصل البيانات.

"A process that tries to read from an empty channel waits until data is available. It cannot ask whether data is available before reading and, for example, if there is no data, decide not to read that channel."

- العملية التي تقرأ من قناة فارغة تنتظر حتى تتوفر البيانات.
- **الأهم :**العملية لا يمكنها أن تسأل "هل توجد بيانات في القناة؟" قبل أن تحاول القراءة. محاولة القراءة نفسها هي الفعل الذي يحدد ما إذا كانت ستنتظر أم لا. لا يوجد فحص مسبق. إذا حاولت القراءة والقناة فارغة، **ستنتظر إجبارياً** .ولا يمكنها أن تقرر "حسناً، لا توجد بيانات الآن، سأفعل شيئاً آخر بدلاً من القراءة". القراءة تُلزمها بالانتظار.
- **DETERMINISM الحتمية :**هذا هو السبب الرئيسي لهذه القواعد الصارمة خاصة القراءة المانعة والـ FIFO في نموذجKPN ، سلوك النظام ونواتجه النهائية **محددة بشكل حتمي** بمعرفة بيانات المدخلات الأولية، بغض النظر عن سرعات العمليات المختلفة أو ترتيب تنفيذها النسبي (طالما أن ترتيب القراءة والكتابة على نفس القناة يتم حسب القواعد). القراءة المانعة تضمن أن العمليات لا تتقدم وتستهلك موارد النظام بطريقة غير متوقعة إذا كانت البيانات التي تحتاجها غير جاهزة بعد. نظام FIFO يضمن أن ترتيب معالجة البيانات ثابت. هذان العنصران معاً يؤديان إلى الحتمية، وهي خاصية مرغوبة جداً في تصميم الأنظمة المدمجة، خصوصاً الأنظمة التي تتطلب موثوقية عالية وأداء يمكن التنبؤ به، مثل أنظمة التحكم في الروبوتات. فنحن نريد أن يتصرف الروبوت بنفس الطريقة بالضبط في كل مرة يتعرض فيها لنفس المدخلات البيئية.

لماذا هذا مهم للروبوتات؟

- **التنبؤ بالسلوك** :في الروبوتات، تحتاج إلى أنظمة تتصرف بشكل يمكن التنبؤ به وموثوق. نموذج KPN بقواعده يساهم في بناء أنظمة حتمية يسهل اختبارها وتصحيح أخطائها.
- **تنسيق المهام** :الروبوت يقوم بمهام متعددة (قراءة حساسات، معالجة بيانات، تخطيط حركة، التحكم في المحركات) قد تعمل بشكل متزامن. يوفر KPN إطاراً لتوصيل هذه المهام (العمليات) ببعضها البعض بطريقة منظمة تضمن تدفق البيانات الصحيح في الترتيب المطلوب.
- **إدارة التدفق** :القراءة المانعة هي شكل بسيط من أشكال مزامنة العمليات. العملية التي تعالج البيانات لا تتقدم قبل وصول البيانات اللازمة، مما يمنع حدوث أخطاء بسبب معالجة بيانات قديمة أو غير موجودة.

باختصار، القواعد الأساسية لشبكات عمليات كان تضمن تدفق البيانات بشكل منظم، وتساهم في جعل سلوك النظام المدمج حتمياً وقابلاً للتنبؤ، وهي خاصية بالغة الأهمية عند تصميم أنظمة تتحكم في كيانات مادية مثل الروبوتات.

Kahn Process Networks

- **Kahn process networks are deterministic:**
  - **For a given sequence of inputs, there is only one possible sequence of outputs (regardless, for example, how long time it takes for a certain computation or communication to finish).**

    Looking only at the specification (and not knowing anything about implementation) you can exactly derive the output sequence corresponding to a given input sequence.

النقطة الجوهرية التي تحدثنا عنها بخصوص شبكات عمليات كان(KPN) هي خاصية **الحتمية(Determinism)** .

هذه الخاصية مهمة جداً في تصميم الأنظمة الموثوقة، وخاصة الأنظمة المدمجة التي غالباً ما تعمل في تطبيقات حرجة مثل الروبوتات والتحكم الصناعي.

"Kahn process networks are deterministic" :شبكات عمليات كان هي **حتمية** .

"For a given sequence of inputs, there is only one possible sequence of outputs (regardless, for example, how long time it takes for a certain computation or communication to finish)."

- هذا هو تعريف الحتمية في سياق KPN . إذا قدمنا لنظام KPN تسلسلاً معيناً وثابتاً من البيانات المدخلة (على القناة I في مثالنا السابق مثلاً)، فإن النظام سينتج دائماً **تسلسلاً وحيداً ومميزاً** من البيانات المخرجة (على القناة O).

- الأهم في هذه النقطة هو **"regardless... how long time it takes..."** :الحتمية في KPN لا تعتمد على التوقيت أو سرعة تنفيذ العمليات أو سرعة نقل البيانات عبر القنوات. سواء نفذت عملية معينة بسرعة فائقة أو ببطء شديد، أو سواء استغرق إرسال توكن وقتاً طويلاً أو قصيراً، طالما أن القواعد الأساسية لـ KPN (القراءة المانعة والكتابة غير المانعة وترتيب FIFO في القنوات) محترمة، فإن تسلسل المخرجات سيكون هو نفسه دائماً لنفس تسلسل المدخلات.

"Looking only at the specification (and not knowing anything about implementation) you can exactly derive the output sequence corresponding to a given input sequence."

- هذه النقطة تعزز فكرة الحتمية وتأثيرها العملي. بما أن سلوك النظام حتمي ويعتمد فقط على تسلسل المدخلات وقواعدKPN ، يمكننا نظرياً (من خلال تحليل وصف العمليات وشبكة القنوات فقط، دون الحاجة لمعرفة تفاصيل التنفيذ على جهاز معين أو سرعات المعالجة) أن **نستنتج بدقة** ما هو تسلسل المخرجات المتوقع لأي تسلسل مدخلات معين.

لماذا هذه الخاصية (الحتمية) مهمة جداً في الأنظمة المدمجة ؟

1. **الموثوقية (Reliability) :** في تطبيقات حرجة مثل التحكم في الروبوتات التي تتفاعل مع البيئة، تحتاج إلى نظام يمكن الوثوق به ليتصرف بنفس الطريقة في كل مرة يواجه فيها نفس الظروف. الحتمية تمنحنا هذا الضمان.
2. **الاختباروتصحيح الأخطاء (Testing and Debugging) :**الأنظمة غير الحتمية (Non-deterministic) يمكن أن تظهر فيها أخطاء يصعب جداً تتبعها، لأن الخطأ قد يحدث في بعض الأحيان ولا يحدث في أحيان أخرى لنفس المدخلات، اعتماداً على عوامل التوقيت الدقيقة. مع نظام حتمي، إذا واجهنا خطأً لتسلسل مدخلات معين، يمكننا إعادة تقديم نفس التسلسل وسنواجه الخطأ نفسه، مما يسهل تحديد سببه وإصلاحه.
3. **التحقق (Verification) :**القدرة على التنبؤ الدقيق بالمخرجات من المدخلات تجعل عملية التحقق من صحة تصميم النظام أسهل وأكثر فعالية. يمكننا مقارنة المخرجات المتوقعة نظرياً بالمخرجات الفعلية للنظام للتأكد من أنه يعمل كما هو مصمم.
4. **الاستقلالية عن التنفيذ (Implementation Independence) :**بما أن الحتمية لا تعتمد على سرعات التنفيذ، يمكن تصميم النظام باستخدام نموذج KPN ثم تنفيذه على منصات أجهزة مختلفة أو باستخدام جدولة مختلفة للعمليات، ومع ذلك نظل متأكدين من أن السلوك الوظيفي (تسلسل المخرجات) لن يتغير.

باختصار، إن القواعد التي رأيناها سابقاً في KPN (القراءة المانعة، FIFO) ليست مجرد تفاصيل عشوائية، بل هي مصممة خصيصاً لضمان خاصية الحتمية. وهذه الخاصية هي جوهر نموذج KPN وتجعله مناسباً جداً لتصميم الأنظمة المدمجة الموثوقة التي تتطلب سلوكاً يمكن التنبؤ به.

Kahn Process Networks

- More on read and write limitations
  - A process cannot wait for data on more than one channel at a time
  - Only a single process is allowed to read from a certain channel
- What if the output data has to be sent to more than one process?
  - Data must be duplicated inside processes
- This limited model of computation implies:
  - More modeling effort for complex systems
  - Retained determinism!

سوف نستعرض هنا بعض **القيود والتبعات** المترتبة على القواعد الصارمة لنموذج Kahn Process Networks (KPN) التي تحدثنا عنها (خاصة طبيعة القراءة المانعة). فهم هذه القيود مهم لنتمكن من استخدام النموذج بشكل صحيح وتقييم مدى ملاءمته لتطبيق معين.

- **"A process cannot wait for data on more than one channel at a time"**
  - إذا كانت تحتاج العملية في KPN للقراءة من عدة قنوات (لديها عدة مدخلات)، يمكنها فقط أن تنتظر (تنحظر) على **قناة واحدة فقط** في لحظة معينة.
  - **ماذا يعني هذا عملياً؟** يعني أن العملية لا يمكنها أن تقول "أنا بحاجة لبيانات من القناة A أو القناة B، وأي بيانات تصل أولاً من أي منهما سأبدأ معالجتها". هي يجب أن تقرر القراءة من A وتنتظرها، أو تقرر القراءة من B وتنتظرها. إذا كانت بحاجة للبيانات من كليهما، يجب أن تقرأ من واحدة أولاً (وتنتظر إذا كانت فارغة)، ثم تقرأ من الثانية (وتنتظر إذا كانت فارغة بعد القراءة الأولى). لا يوجد انتظار "مشترك" أو "اختياري" بين قنوات متعددة.
  - **في مجال الروبوتات** :بفرض لدينا عملية في الروبوت تحتاج بيانات من مستشعر مسافة ومن كاميرا لتحديد موقعها. إذا كانت هذه العملية مصممة كعملية واحدة في KPN ، فقد تواجه صعوبة في معالجة البيانات فور وصولها من أي من المستشعرين. قد نحتاج إلى تصميم العملية بطريقة معينة أو تقسيمها إلى عمليات فرعية لتجاوز هذا القيد.
- **"Only a single process is allowed to read from a certain channel"**
  - قناة KPN هي لقارئ واحد فقط. بمجرد أن تُرسل البيانات إلى قناة معينة، هناك **عملية واحدة فقط** مسموح لها بقراءة هذه البيانات من هذه القناة. لا يمكن لعمليتين مختلفتين أن تقرءان نفس التوكن

من نفس القناة. إذا حاولت عمليتان القراءة من نفس القناة، فإن سلوك النظام يصبح غير محدد وقد يحدث خطأ.

- **في مجال الروبوتات** :إذا كان لدينا بيانات معينة (مثل قراءة بوصلة) يحتاجها أكثر من جزء في نظام الروبوت (مثلاً، عملية لتحديد الاتجاه وأخرى لرسم الخريطة)، لا يمكننا ببساطة توصيل مخرج عملية قراءة البوصلة بقناة واحدة ثم جعل عمليتي "تحديد الاتجاه" و"رسم الخريطة" تقرآن من نفس القناة.

: "What if the output data has to be sent to more than one process?"

هذا السؤال يطرح حلاً للتحدي الأخير الذي ذكرناه (عدم إمكانية قراءة قناة بواسطة عمليتين).

- **"Data must be duplicated inside processes"**

- الحل في KPN هو أنه إذا كانت بيانات معينة يجب أن تصل إلى أكثر من عملية واحدة، فإن العملية التي تنتج هذه البيانات يجب أن تقوم **بتكرار (Duplication)** هذه البيانات وإرسال نسخة منها عبر **قناة منفصلة** لكل عملية مستقبلة.
- **مثال توضيحي** :عملية قراءة البوصلة (Process_Compass) تقرأ القيمة. إذا كانت هذه القيمة مطلوبة لعملية تحديد الاتجاه (Process_Heading) وعملية رسم الخريطة(Process_Mapping) ، فإن Process_Compass يجب أن ترسل القيمة عبر قناة C_to_Heading إلىProcess_Heading ، وتُكرر القيمة ثم ترسل نسخة أخرى منها عبر قناة C_to_Mapping إلى Process_Mapping. لا يمكن لـProcess_Heading و Process_Mapping القراءة من نفس القناة الخارجة من Process_Compass.
- **تأثيره** :هذا يضيف بعض التعقيد وجهد النمذجة، حيث تحتاج العمليات المنتجة للبيانات إلى معرفة من هم المستقبلون وتقوم بتكرار البيانات وإرسالها عبر قنوات مختلفة.

تبعات هذه القيود التي ذكرناها (خاصة قيود القراءة والكتابة) :

- **"More modeling effort for complex systems"**

- بسبب الحاجة إلى تجنب الانتظار على قنوات متعددة وتكرار البيانات للمستقبلين المتعددين، فإن بناء نماذج KPN لأنظمة معقدة قد يتطلب جهداً أكبر في تصميم هيكل العمليات والقنوات لضمان التدفق الصحيح للبيانات وتجنب الجمود (deadlock) الذي قد يحدث بسبب الانتظار.
- تصميم شبكة KPN لنظام مدمج معقد يتطلب تخطيطاً دقيقاً لكيفية تدفق البيانات بين مختلف الوحدات الوظيفية.

- **"Retained determinism!"**

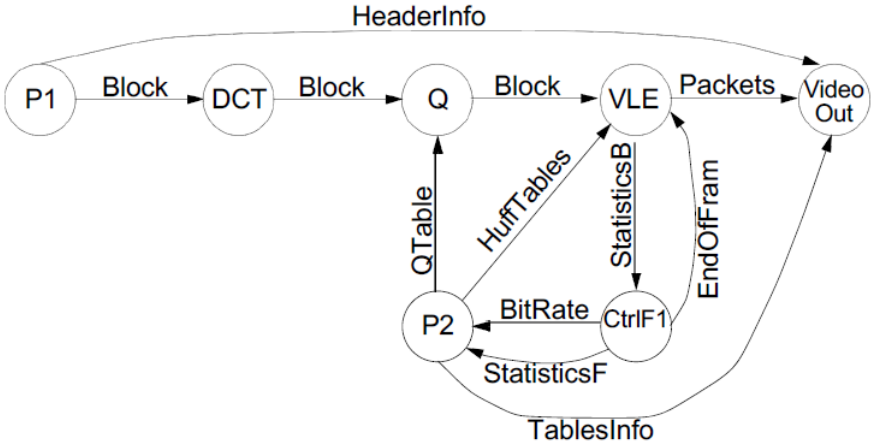
- هذه هي الخلاصة الإيجابية! على الرغم من هذه القيود التي قد تبدو معقدة، فإن الالتزام بقواعد KPN**يُحافظ** على خاصية الحتمية (Determinism) التي تحدثنا عنها سابقاً. هذه القيود هي جزء من الثمن الذي ندفعه للحصول على نظام يمكن التنبؤ بسلوكه.

نموذجKPN ، على الرغم من أنه يمنحنا خاصية الحتمية القوية، يأتي مع بعض القيود في طريقة تفاعل العمليات (لا انتظار متعدد القنوات، قناة واحدة لقارئ واحد). هذه القيود تتطلب جهداً إضافياً عند نمذجة الأنظمة المعقدة (مثل الحاجة لتكرار البيانات للمستقبلين المتعددين)، ولكن هذا الجهد الإضافي يضمن بقاء النظام حتمياً، وهي خاصية بالغة الأهمية للأنظمة المدمجة الموثوقة والروبوتات.

فهمنا لهذه القيود سيساعدنا على تصميم أنظمة KPN بشكل صحيح وتجنب المشاكل المحتملة.

Kahn Process Networks: an Example

KPN model of encoder for Motion JPEG (M-JPEG) video compression format:



نقدم هنا **نموذج KPN لمشفّر (encoder) لصيغة ضغط الفيديو Motion JPEG (M-JPEG)** و هو تطبيق عملي ومثال واضح للمفاهيم النظرية التي ناقشناها حتى الآن حول Kahn Process Networks (KPN) .

إن معالجة الفيديو والصور هي جزء أساسي في كثير من أنظمة الروبوتات (مثل رؤية الحاسوب، التعرف على الأشياء، الملاحة المرئية). فهم كيفية عمل مشفر الفيديو باستخدام نموذج مثل KPN يعطينا رؤية قيمة حول كيفية تنظيم هذه المهام المعقدة في نظام مدمج.

فيما يلي سوف نحلل هذا النموذج خطوة بخطوة:

M-JPEG هي صيغة تضغط كل إطار فيديو بشكل مستقل باستخدام خوارزمية JPEG لضغط الصور.

الرسم البياني لشبكة العمليات والقنوات :نرى العمليات (الدوائر) والقنوات (الأسهم) التي تشكل نموذج KPN لمشفّر M-JPEG. لنتبع تدفق البيانات الرئيسي:

- **P1** :هذه هي العملية الأولى، تمثل مرحلة ما قبل المعالجة أو تقسيم الصورة. فيM-JPEG ، غالباً ما تُقسم الصورة (الإطار) إلى كتل (Blocks) صغيرة (عادة 8 x 8بكسل).
  - المدخلات: تأتي من مصدر خارجي (غير ممثل هنا بوضوح كقناة I كما في المثال النظري، لكن يمكننا تخيل أنها البيانات الأولية للإطار).

- المخرجات: ترسل البيانات في شكل كتل (Blocks) إلى العملية التالية عبر القناة المسماة Block

- **DCT :**هذه العملية تقوم بتحويل جيب التمام المتقطع (Discrete Cosine Transform - DCT) . هذه خطوة أساسية في ضغط JPEG حيث يتم تحويل بيانات البكسل في كل كتلة إلى مجموعة من المعاملات الترددية.
  - المدخلات: تستقبل الكتل (Blocks) من القناة Block
  - المخرجات: ترسل المعاملات المحسوبة إلى العملية التاليةQ
- **Q :**هذه العملية تقوم بعملية التكميم (Quantization) .في هذه الخطوة يتم تقريب المعاملات الترددية إلى قيم محددة باستخدام جدول التكميم(QTable) ، وهذا هو الجزء الذي يؤدي إلى فقدان بعض المعلومات (ضغط البيانات).
  - المدخلات: تستقبل المعاملات من العمليةDCT ، وتستقبل أيضاً جدول التكميم (QTable) من العملية P2 عبر القناة المسماة QTable. حيث نلاحظ أن QTable سهمها يتجه من P2 إلىQ ، مما يعني أن P2 هي التي تنتج جدول التكميم هذا (ربما بناءً على مستوى الجودة المطلوب).
  - المخرجات: ترسل الكتل المكمّمة (Blocks) إلى العملية VLE عبر القناة المسماة Block (اسم القناة هو نفسه، لكنها تحمل بيانات مختلفة الآن).
- **VLE :**هذه العملية تقوم بالترميز ذي الطول المتغير(Variable-Length Encoding - VLE) ، وأشهر مثال عليه هو ترميز هوفمان (Huffman Encoding) . في هذه الخطوة، يتم تمثيل المعاملات المكمّمة باستخدام عدد متغير من البتات، بحيث يتم استخدام عدد أقل من البتات للقيم الأكثر تكراراً، مما يؤدي إلى المزيد من الضغط.
- **Video, Out** : هذه العملية هي المسؤولة عن تجميع البيانات المرمّزة في حزم(Packets) ، وإضافة معلومات رأس الإطار (HeaderInfo) التي تأتي مباشرة من P1 (ربما معلومات عن أبعاد الصورة، نوع الضغط، إلخ). ثم تقوم بإخراج تيار البيانات المضغوط.
  - المدخلات: تستقبل الحزم (Packets) منVLE ، ومعلومات الرأس (HeaderInfo) منP1 ، وإشارة نهاية الإطار (EndOfFram) من.VLE
  - المخرجات: تيار الفيديو المضغوط إلى الخارج) غير ممثل هنا كقناةO ، لكنه المخرج النهائي للنظام.(

تحليل النموذج في سياق KPN

- نرى بوضوح العمليات (الدوائر) والقنوات (الأسهم).
- الأسهم هي قنوات أحادية الاتجاه.(unidirectional)
- البيانات تتدفق ك "توكنات" (كتل، معاملات، جداول، حزم).
- هذا النموذج يفترض أن العمليات تقرأ بشكل مانع (blocking read) وتكتب بشكل غير مانع non-blocking writeعلى هذه القنوات لضمان الحتمية التي تحدثنا عنها.

- نلاحظ كيف يتم تداول المعلومات الإحصائية وجداول التحكم مثل QTable و HuffTables بين العمليات، مما يسمح لـ P2 بتعديل سلوك عمليتي Q و VLE ديناميكياً للتحكم في جودة الضغط ومعدل البتات. هذا مثال على التغذية الراجعة (feedback) في نموذجKPN

أهمية هذا المثال في مجال الروبوتات:

- يوضح كيف يمكن تقسيم مهمة معقدة (مثل ضغط الفيديو) إلى مجموعة من العمليات المترابطة.
- يظهر كيف يتم تدفق البيانات بين هذه العمليات في نظام تدفق بيانات.
- يقدم مثالاً عملياً على كيفية تطبيق نموذج KPN في نظام مدمج حقيقي.
- يعطي فكرة عن المكونات الأساسية (من منظور معالجة البيانات) لأحد أنواع مشفرات الفيديو التي قد نجدها في أنظمة رؤية الروبوتات.

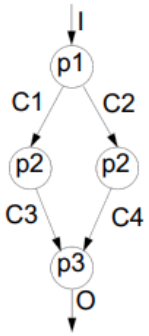
هذا المثال يربط كل المفاهيم التي تعلمناها عن KPN :العمليات، القنوات، تدفق البيانات، وحتى فكرة التغذية الراجعة (Feedback)لتعديل سلوك النظام.

Kahn Process Networks: a Simpler Example

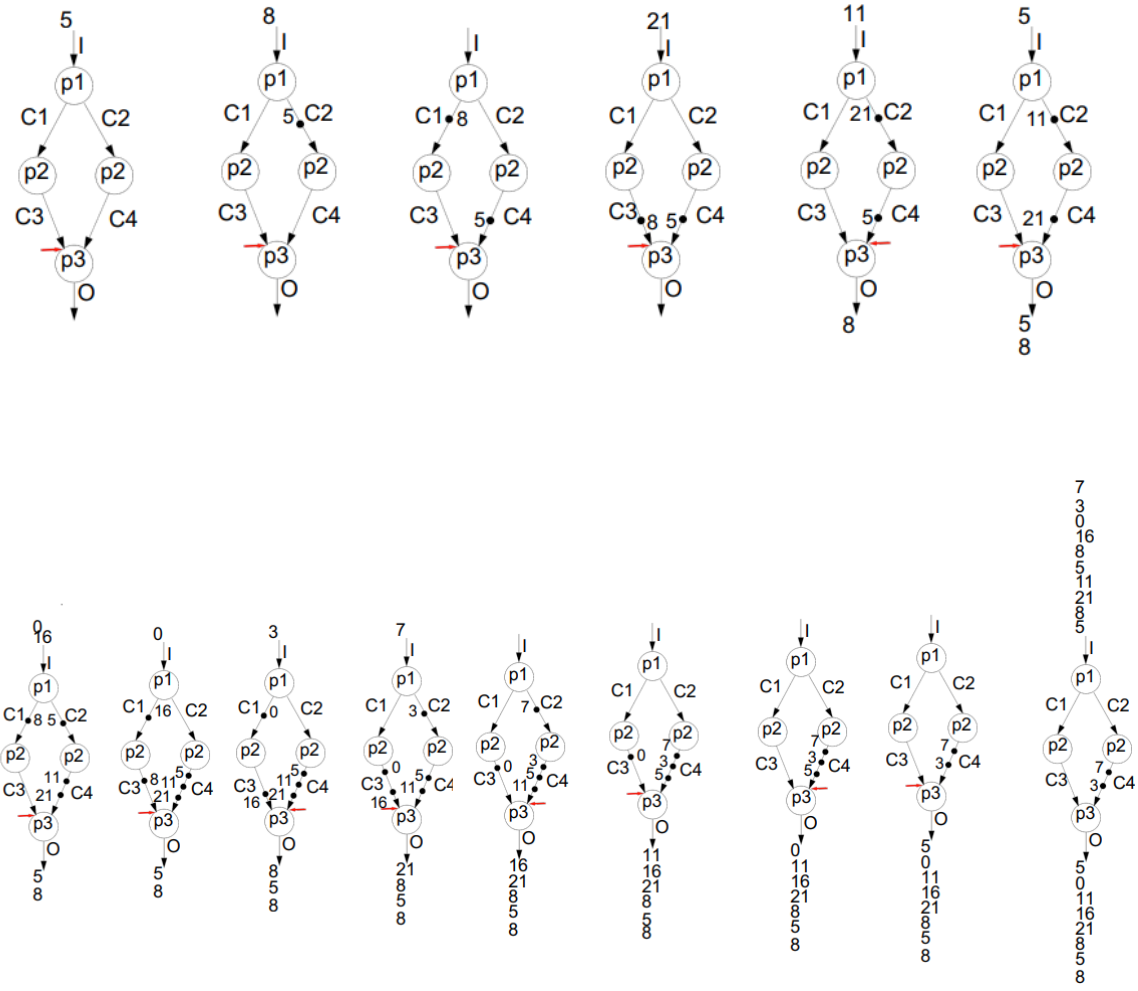
```
Process p1( in int a, out int x, out int y) {
int k;
loop
    k = a.receive();
    if k mod 2 = 0 then
            x.send(k);
    else
            y.send(k);
    end if;
end loop; }
```

```
Process p2( in int a, out int x) {
int k;
loop
    k = a.receive();
    x.send(k);
end loop; }
```

```
Process p3(in int a, in int b, out int x) {
int k; bool sw = true;
loop
    if sw then
            k = a.receive();
    else
            k = b.receive();
    end if;
    x.send(k);
    sw = !sw;
end loop; }
```




```
channel int I, O, C1, C2, C3, C4;
p1(I, C1, C2);
p2(C1, C3);
p2(C2, C4);
p3(C3, C4, O);
```


I
p1
C1
C2
p2
C3
C4
p3
O

Kahn Process Networks: Determinism

- For the same input sequence, the produced output sequence is always the same
- These factors entirely determine the outputs of the system:
  - Processes
  - The network
  - Initial tokens
- Timing of the processes and channels do not affect the outputs of the system

هنا نؤكد على مفهوم **الحتمية (Determinism)** في Kahn Process Networks (KPN) ، وهي النقطة التي تحدثنا عنها سابقاً وشددنا على أهميتها.

"For the same input sequence, the produced output sequence is always the same" :هذا هو التعريف الأساسي للحتمية في KPN. إذا قدمنا لنظام KPN نفس سلسلة المدخلات بالضبط في كل مرة، فإننا سنحصل دائماً على نفس سلسلة المخرجات بالضبط. لا مجال للصدفة أو التباين في النتيجة النهائية.

"These factors entirely determine the outputs of the system" :

هذه النقطة تحدد العوامل الوحيدة التي تؤثر وتحدد تسلسل المخرجات في نظام KPN. ما هي هذه العوامل؟

- **Processes** :منطق وسلوك العمليات نفسها (ماذا تفعل العمليات بالبيانات التي تقرأها وماذا تنتج).
- **The network**: هيكل الشبكة (كيف تتصل العمليات ببعضها البعض عبر القنوات).
- **Initial tokens** :أي توكنات بيانات قد تكون موجودة في القنوات عند بدء تشغيل النظام.

هذه العوامل الثلاثة (العمليات، هيكل الشبكة، والتوكنات الأولية) هي كل ما نحتاجه للتنبؤ بسلوك النظام ومخرجاته بشكل كامل.

"Timing of the processes and channels do not affect the outputs of the system" :

هذه هي النتيجة المباشرة والأكثر إثارة للاهتمام للحتمية في KPN سرعة تنفيذ العمليات(كم من الوقت تستغرق عملية p1أو p2 أو p3 لتنفيذ خطوتها الحسابية) وسرعة نقل البيانات عبر القنوات **لا تؤثران على تسلسل المخرجات النهائية** . قد تؤثران على توقيت ظهور المخرجات أو السرعة الكلية للنظام، ولكن ليس على ما هي القيم التي ستخرج وبأي ترتيب.

في مجال الروبوتات؟

كما ذكرنا سابقاً، الحتمية ضرورية جداً عند تصميم أنظمة التحكم، خصوصاً في الروبوتات التي تحتاج للتفاعل مع العالم المادي بشكل موثوق ويمكن التنبؤ به. لنتخيل أن سلوك نظام تحديد المواقع في روبوت ما يتغير عشوائياً في كل

مرة لنفس قراءات المستشعرات بسبب عوامل توقيت غير محددة. هذا سيكون كارثياً. نموذج KPN يوفر إطاراً يساعد في بناء أنظمة تتجنب هذا السلوك غير المرغوب فيه، مما يزيد من موثوقية وأمان الروبوت.