

Class 2 Outline

1. Monte-Carlo:

- **Framework/Definition**
- **Algorithm**
- **Examples**

2. Random Number Generation

Monte-Carlo Framework

Estimate $\theta = E[h(\mathbf{X})]$

where $\mathbf{X} = \{X_1, \dots, X_m\}$ is a random vector in \mathbb{R}^m ,

$h(\cdot)$ is a function $\mathbb{R}^m \rightarrow \mathbb{R}$,

and $E[| h(\mathbf{X}) |] < \infty$

Monte-Carlo Algorithm

1. Generate n samples of \mathbf{X} : $\mathbf{X}^1, \dots, \mathbf{X}^n$
2. Compute $h(\mathbf{X}^1), h(\mathbf{X}^2), \dots, h(\mathbf{X}^n)$
3. Estimate $\theta = E[h(\mathbf{X})]$ with
$$\bar{\theta} = [h(\mathbf{X}^1)+h(\mathbf{X}^2)+\dots+h(\mathbf{X}^n)]/n$$

- Why is $\bar{\theta}$ a good estimator?

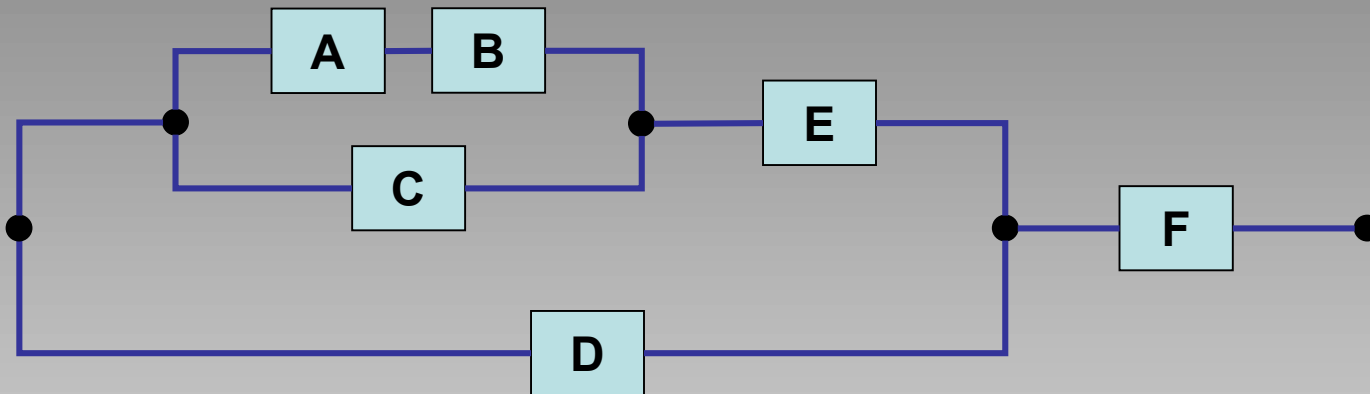
A Sales Incentive Plan

A company has a sales incentive plan with the following structure: In the months when sales are lower than \$10,000, a salesrep get a base salary of \$2000. If the monthly sales achieved are between \$10,000 and \$15,000, then he/she is paid 20% of the sales. Finally, if the monthly sales figure is larger than \$15,000, the salesrep get 30% of the sales.

Assuming the monthly sales achieved by a salesman follows an exponential distribution with mean \$15,000, what is the mean and standard deviation of the salesman's monthly salary under this compensation plan?

Product Reliability

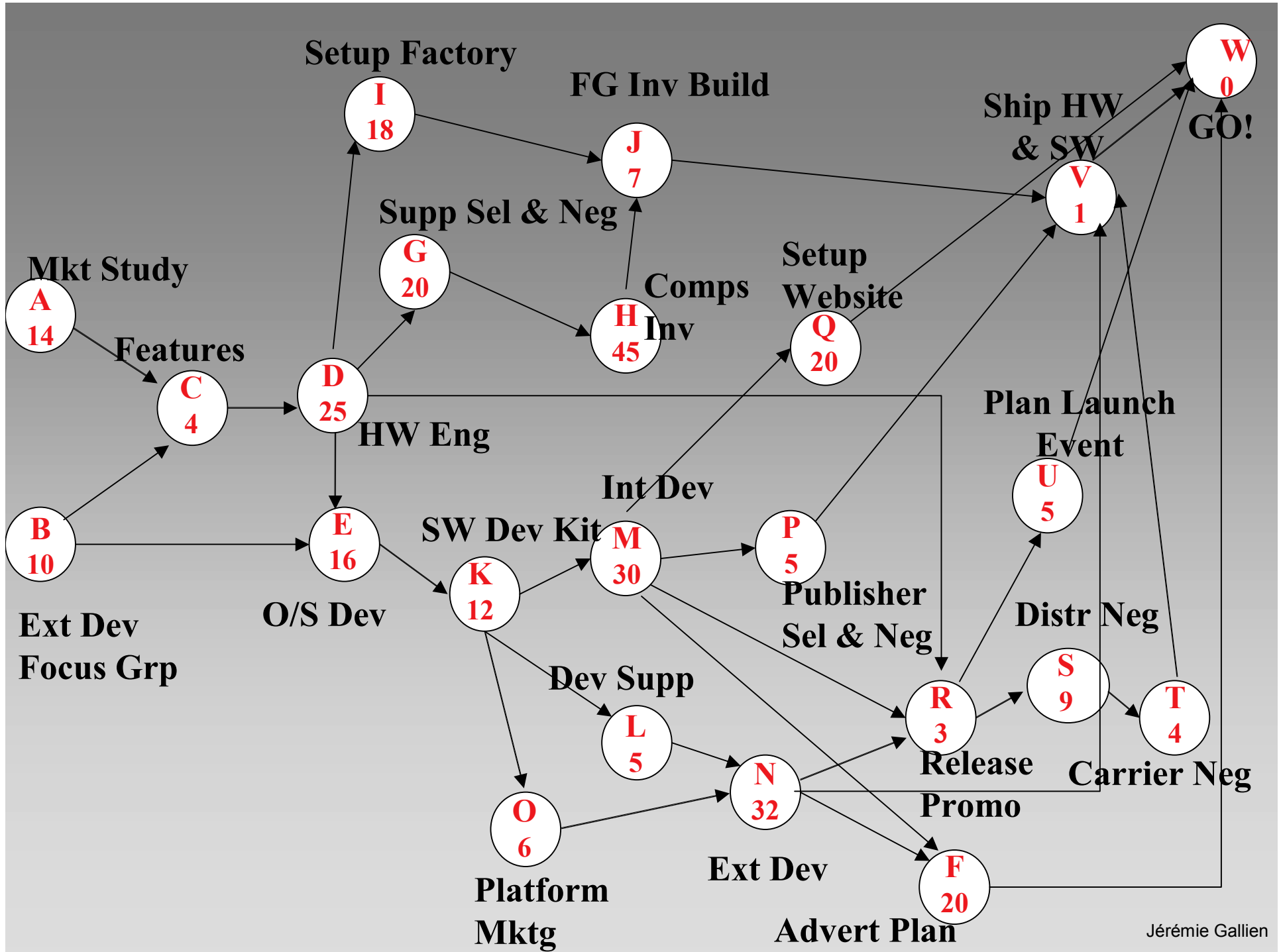
- Consider a product reliability model given by:



- Component life lengths: $X_A \sim \exp(5.1)$; $X_B \sim N(4.5, 1)$; $X_C \sim \exp(6.5)$
 $X_D \sim \exp(5.4)$; $X_E \sim \exp(6.4)$; $X_F \sim N(5.5, 0.8)$
- What is the probability that the product will function for at least 3 years (warranty period)?*

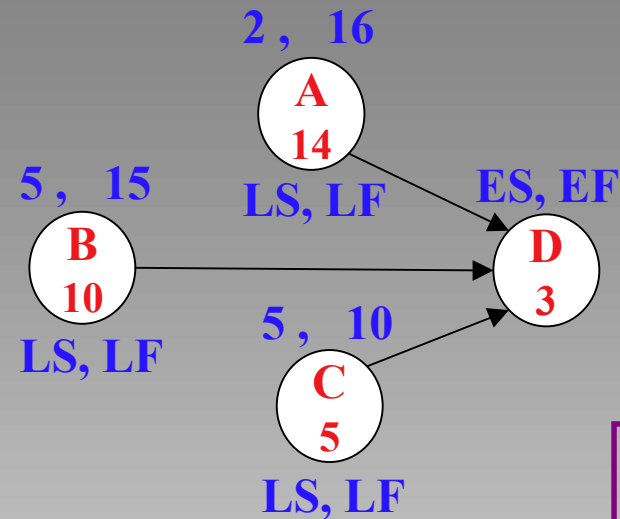
DreamCast Launch Project

Task	Description	Duration (weeks)	Precedence
A	Market Study	14	
B	External Developer Focus Group	10	
C	Feature Selection	4	A,B
D	Hardware Engineering	25	C
E	Operating System Devlpt.	16	D,B
F	Advertising Campaign	20	D,M,N
G	Supplier Selection & Negotiation	20	D
H	Component Inventory Buildup	45	G
I	Assembly Facility Setup	18	D
J	Finished Good Inventory Buildup	7	I,H
K	Library & Programmer Toolkit Devlpt.	12	E
L	External Development Support Setup	5	K
M	Internal Game Devlpt.	30	K
N	External Game Devlpt.	32	L,O
O	Platform Promotion	6	K
P	Publisher Selection & Negotiation	5	M
Q	Website Setup	20	M
R	Release Promotion Material Design	3	D,M,N
S	Distribution Channels Devlpt. & Negotiation	9	R
T	Carrier Selection & Negotiation	4	S
U	Launch Event Organization & PR	5	R
V	Hardware & Software Shipment	1	J,T,P,N
W	Launch!	0	V,U,Q,F

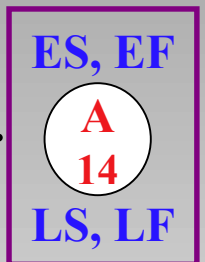
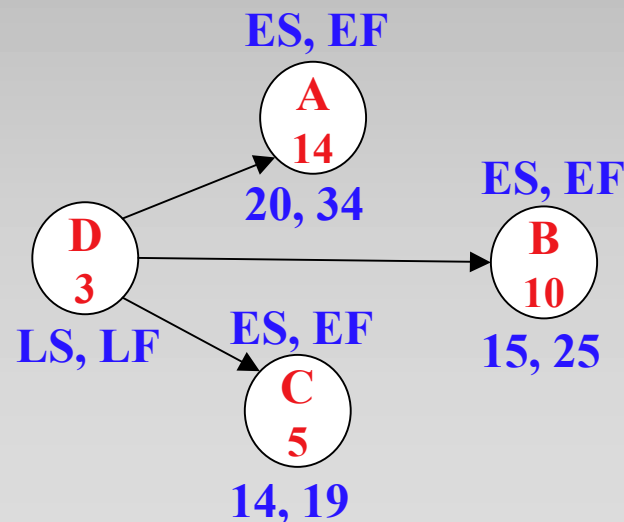


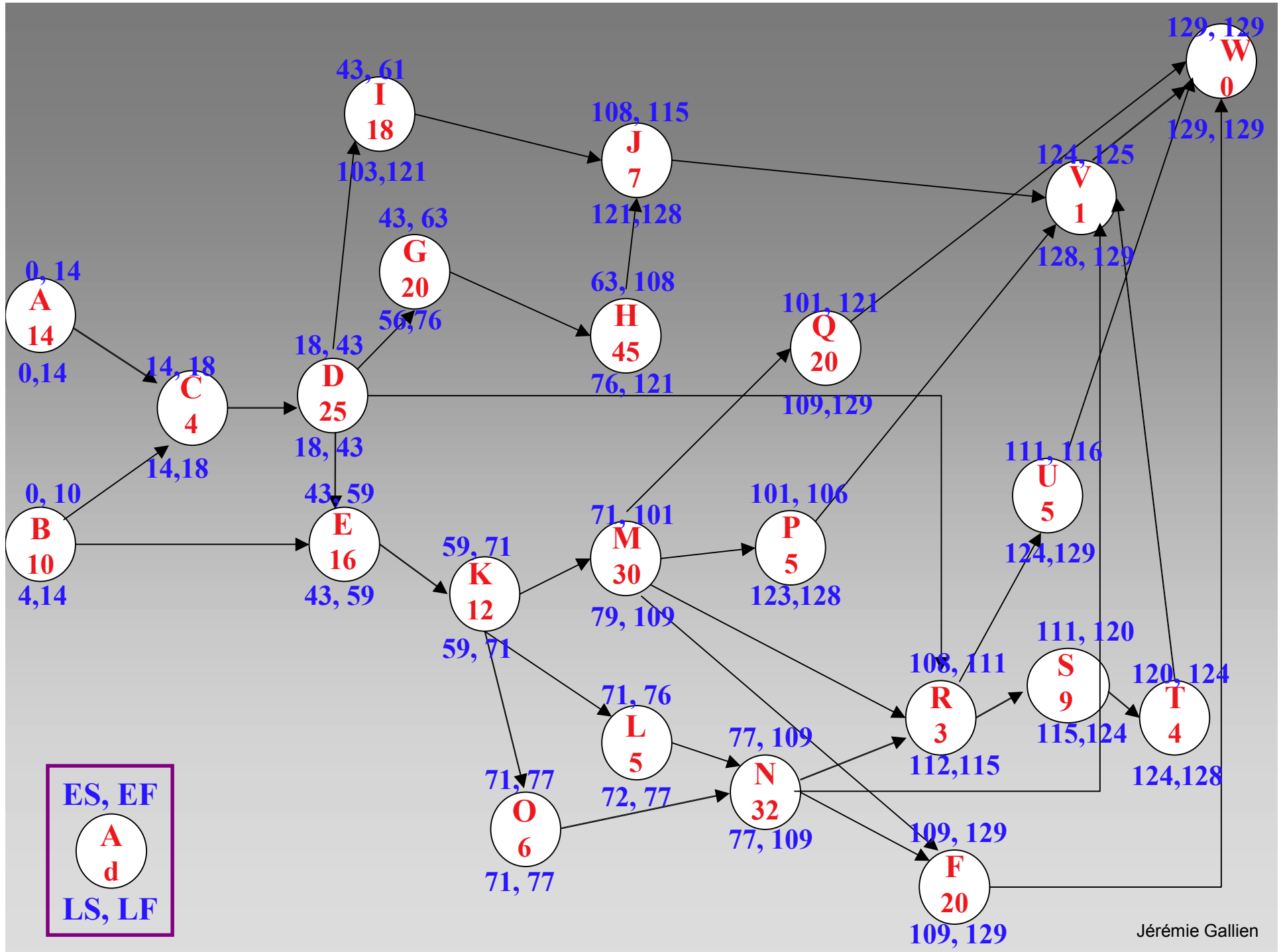
Critical Path Algorithm

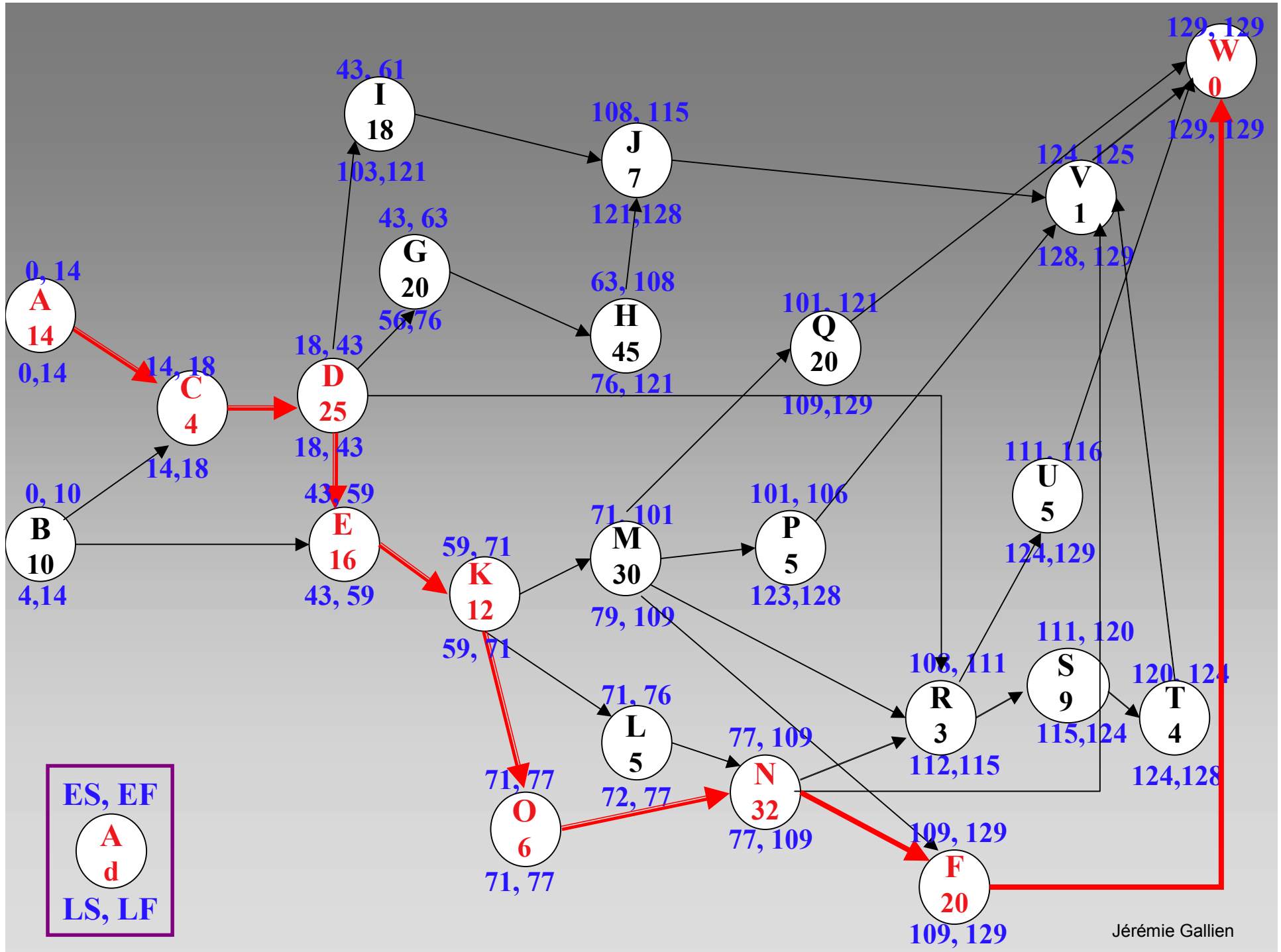
1. A task Early Start Date is the largest of all Early Finish Dates of its predecessors



2. A task Late Finish Date is the earliest of all Late Start Dates of its successors







Critical Path Solution

Task	Description	Duration (weeks)	Precedence	LS	LF	ES	EF	Slack
A	Market Study	14		-129	-115	-129	-115	0
B	External Developer Focus Group	10		-125	-115	-129	-119	4
C	Feature Selection	4	A,B	-115	-111	-115	-111	0
D	Hardware Engineering	25	C	-111	-86	-111	-86	0
E	Operating System Devt.	16	D,B	-86	-70	-86	-70	0
F	Advertising Campaign	20	D,M,N	-20	0	-20	0	0
G	Supplier Selection & Negotiation	20	D	-73	-53	-86	-66	13
H	Component Engineering & Inventory Buildup	45	G	-53	-8	-66	-21	13
I	Assembly Facility Setup	18	D	-26	-8	-86	-68	60
J	Finished Good Inventory Buildup	7	I,H	-8	-1	-21	-14	13
K	Library & Programmer Toolkit Devt.	12	E	-70	-58	-70	-58	0
L	External Development Support Setup	5	K	-57	-52	-58	-53	1
M	Internal Game Devt.	30	K	-50	-20	-58	-28	8
N	External Game Devt.	32	L,O	-52	-20	-52	-20	0
O	Platform Promotion	6	K	-58	-52	-58	-52	0
P	Publisher Selection & Negotiation	5	M	-6	-1	-28	-23	22
Q	Website Setup	20	M	-20	0	-28	-8	8
R	Release Promotion Material Design	3	D,M,N	-17	-14	-20	-17	3
S	Distribution Channels Devt. & Negotiation	9	R	-14	-5	-17	-8	3
T	Carrier Selection & Negotiation	4	S	-5	-1	-8	-4	3
U	Launch Event Organization & PR	5	R	-5	0	-17	-12	12
V	Hardware & Software Shipment	1	J,T,P,N	-1	0	-4	-3	3
W	Launch!	0	V,U,Q,F	0	0	0	0	0

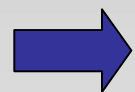
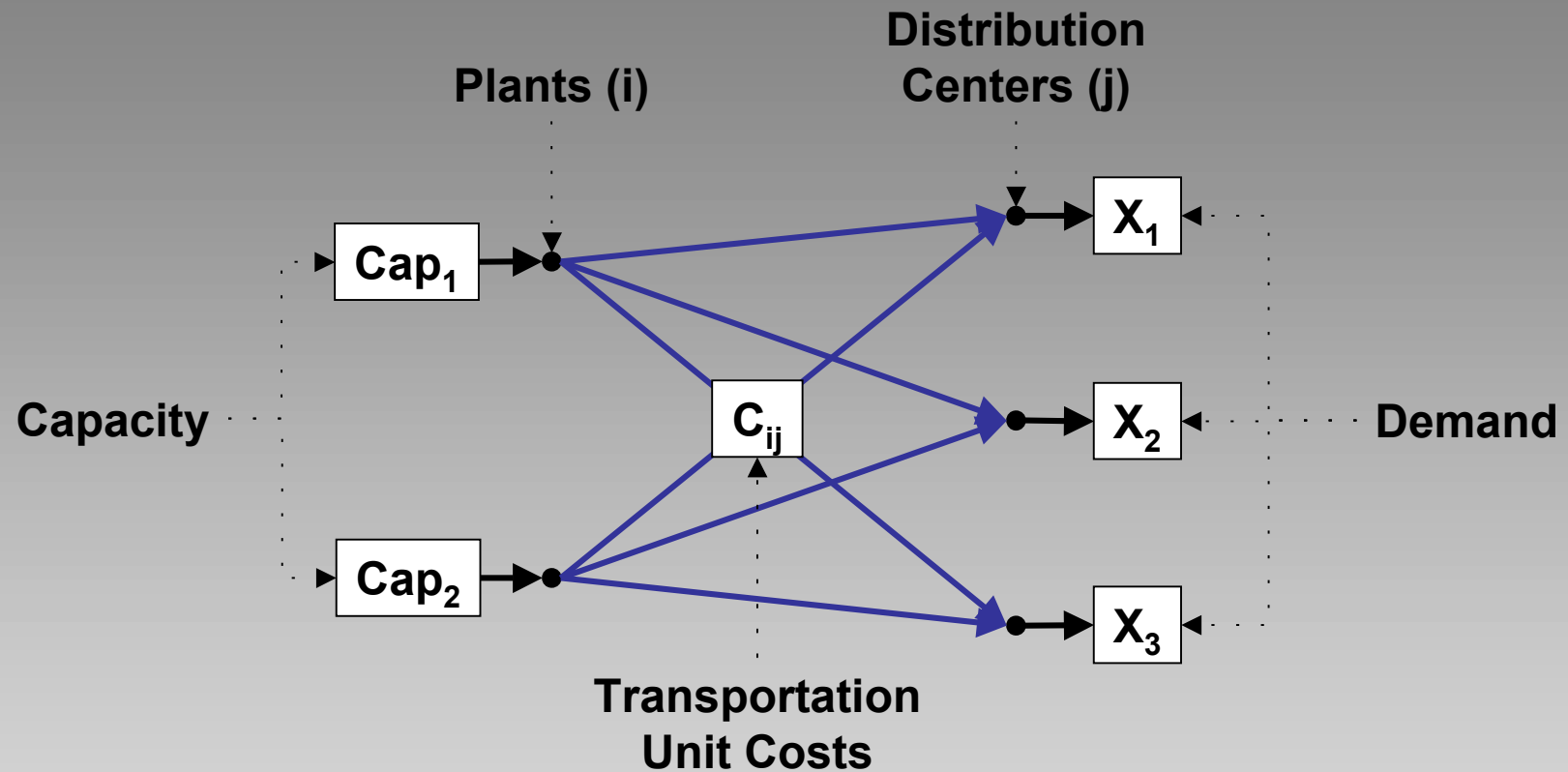
Stochastic Task Durations

Task	Description	Duration (weeks)	Precedence	LS	LF	ES	EF	Slack
A	Market Study	14		-129	-115	-129	-115	0
B	External Developer Focus Group	10		-125	-115	-129	-119	4
C	Feature Selection	4	A,B	-115	-111	-115	-111	0
D	Hardware Engineering	25	C	-111	-86	-111	-86	0
E	Operating System Devt.	16	D,B	-86	-70	-86	-70	0
F	Advertising Campaign	20	D,M,N	-20	0	-20	0	0
G	Supplier Selection & Negotiation	20	D	-73	-53	-86	-66	13
H	Component Engineering & Inventory Buildup	45	G	-53	-8	-66	-21	13
I	Assembly Facility Setup	18	D	-26	-8	-86	-68	60
J	Finished Good Inventory Buildup	7	I,H	-8	-1	-21	-14	13
K	Library & Programmer Toolkit Devt.	12	E	-70	-58	-70	-58	0
L	External Development Support Setup	5	K	-57	-52	-58	-53	1
M	Internal Game Devt.	30	K	-50	-20	-58	-28	8
N	External Game Devt.	32	L,O	-52	-20	-52	-20	0
O	Platform Promotion	6	K	-58	-52	-58	-52	0
P	Publisher Selection & Negotiation							
Q	Website Setup							
R	Release Promotion Material Design							
S	Distribution Channels Devt. & Negotiation							
T	Carrier Selection & Negotiation							
U	Launch Event Organization & PR							
V	Hardware & Software Shipment							
W	Launch!							

Duration of development tasks E, M and N now follow exponential distributions with means 16, 30 and 32 respectively.

What is the probability that the total project duration will be more than 135 weeks?

Stochastic Programming



Stochastic Demand

$$\left\{ \begin{array}{l} X_1 \sim N(1500, 350) \\ X_2 \sim N(2000, 600) \\ X_3 \sim N(1500, 450) \end{array} \right.$$

Random Number Generation

- The key in a Monte-Carlo simulation is to generate sample values drawn from a known probability distribution. How is this done?

Generate Sample Value from Uniform[0,1]

```
graph TD; A[Generate Sample Value from Uniform[0,1]] --> B[Discrete Distribution]; A --> C[Continuous Distribution];
```

Discrete Distribution

- Inverse transform method
- Rejection method

Continuous Distribution


- Inverse transform method
- Rejection method
- Composition method
- Polar method

U[0,1] Sample Generation

1. Calculate Z_1, Z_2, \dots given by the Linear Congruential Generator:

$$Z_i = (aZ_{i-1} + c) \text{ modulo } m$$

where m, a, c and Z_0 are non-neg. integers


modulus


seed

2. Take $U_i = Z_i / m$ as a proxy to an independent sequence of U[0,1] random variates (note: $0 \leq U_i \leq 1$ for all i)

Linear Congruential Generator

- Example: $Z_i = (11Z_{i-1}) \text{ modulo } 16$

$$Z_0 = 1$$

$$Z_1 = (11) \text{ mod } 16 = 11$$

$$Z_2 = (121) \text{ mod } 16 = 9$$

$$Z_3 = (99) \text{ mod } 16 = ?$$

$$Z_4 = (33) \text{ mod } 16 = ?$$

- *Huge* literature on LCG periods, parameter choice, statistical tests for randomness...
- Crystal Ball uses $Z_i = (630,360,016 Z_{i-1}) \text{ mod } (2^{31}-1)$

Discrete Distributions

- We want to simulate a random variable X defined by $P(X = x_j) = p_j$, $j=0,1,\dots,m$

1. Generate a sample U from $U[0,1]$
2. if $U < p_0$ then $X = x_0$
if $p_0 < U < p_0 + p_1$ then $X = x_1$
.....
if $p_0 + \dots + p_{j-1} < U < p_0 + \dots + p_j$ then $X = x_j$

- Why does this work?

Solution

1. Generate a sample U from $U[0,1]$
2. if $U < p_0$ then $X = x_0$
if $p_0 < U < p_0 + p_1$ then $X = x_1$
.....
if $p_0 + \dots + p_{j-1} < U < p_0 + \dots + p_j$ then $X = x_j$

- $$P(X = x_j) = P(p_0 + \dots + p_{j-1} < U < p_0 + \dots + p_j)$$
$$= p_j$$

Continuous Distributions

- We want to simulate a random variable X defined by $P(X < t) = F(t)$ (F inversible)

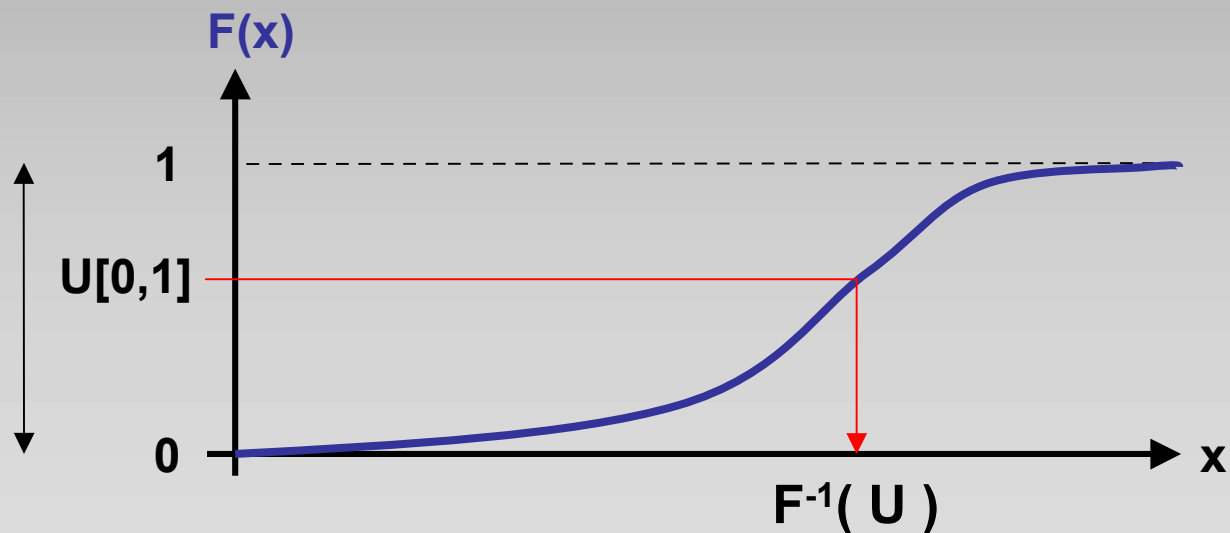
1. Generate a sample U from $U[0,1]$
2. Set $X = F^{-1}(U)$

- Why does this work?

Solution

1. Generate a sample U from $U[0,1]$
2. Set $X = F^{-1}(U)$

- $P (X < t) = P(F^{-1}(U) < t) = P(U < F(t)) = F(t)$



Exponential Distribution

- Variable $\exp(\lambda)$ has cdf. $F(t) = 1 - \exp(-\lambda t)$;
 $F^{-1}(y) = -\log(1-y) / \lambda$

1. Generate U from $U[0,1]$
2. Compute $X = -\log(1-U) / \lambda$
3. The distribution of X is $\exp(\lambda)$!

Class 2 Wrap-Up

- 1. Monte-Carlo Framework, Algorithm & Examples**
- 2. Crystal Ball Modeling**
- 3. Theory of Random Number Generation**