

第二部分 分布式算法

汪炆

第二次课

中国科学技术大学计算机系
国家高性能计算中心（合肥）

§ 2.1.1 系统

2.异步系统

- 异步：msg传递的时间和一个处理器的两个相继步骤之间的时间无固定上界

例如，Internet中，email虽然常常是几秒钟到达，但也可能要数天到达。当然msg延迟有上界，但它可能很大，且随时间而改变。

因此异步算法设计时，须使之独立于特殊的计时参数，不能依赖于该上界。

- 执行片断

一个异步msg传递系统的一个执行片断 α 是一个有限或无限的序列：

$$C_0, \Phi_1, C_1, \Phi_2, C_2, \Phi_3, \dots, (C_0 \text{不一定是初始配置})$$

这里 C_k 是一个配置， Φ_k 是一个事件。若 α 是有限的，则它须结束于某个配置，且须满足下述条件：

§ 2.1.1 系统

❖ 若 $\Phi_k = \text{del}(i,j,m)$ ，则 m 必是 C_{k-1} 里的 $\text{outbuf}_i[l]$ 的一个元素，这里 l 是 p_i 的信道 $\{p_i, p_j\}$ 的标号

从 C_{k-1} 到 C_k 的唯一变化是将 m 从 C_{k-1} 里的 $\text{outbuf}_i[l]$ 中删去，并将其加入到 C_k 里的 $\text{inbuf}_j[h]$ 中， h 是 p_j 的信道 $\{p_i, p_j\}$ 的标号。

即：传递事件将 msg 从发送者的输出缓冲区移至接收者的输入缓冲区。

❖ 若 $\Phi_k = \text{comp}(i)$ ，则从 C_{k-1} 到 C_k 的变化是

① 改变状态：转换函数在 p_i 的可访问状态(在配置 C_{k-1} 里)上进行操作，清空 $\text{inbuf}_i[l]$ ， $(1 \leq l \leq r)$

② 发送 msg ：将转换函数指定的消息集合加到 C_k 里的变量 outbuf_i 上。(Note: 发送 send ，传递 delivery 之区别)

即： p_i 以当前状态(在 C_{k-1} 中)为基础按转换函数改变状态并发出 msg 。

§ 2.1.1 系统

- 执行：一个执行是一个执行片断 $C_0, \Phi_1, C_1, \Phi_2, \dots$ ，这里 C_0 是一个初始配置。
- 调度：一个调度(或调度片段)总是和执行(或执行片断)联系在一起的，它是执行中的事件序列： Φ_1, Φ_2, \dots 。
并非每个事件序列都是调度。例如， $\text{del}(1,2,m)$ 不是调度，因为此事件之前， p_1 没有步骤发送 $(\text{send})m$ 。
若局部程序是确定的，则执行(或执行片断)就由初始配置 C_0 和调度(或调度片断) σ 唯一确定，可表示为 $\text{exec}(C_0, \sigma)$ 。

§ 2.1.1 系统

■ 容许执行：(满足活跃性条件)

异步系统中，若某个处理器有无限个计算事件，每个发送的msg都最终被传递，则执行称为容许的。

Note: 无限个计算事件是指处理器没有出错，但它不蕴含处理器的局部程序必须包括一个无限循环
非形式地说：一个算法终止是指在某点后转换函数不改变处理器的状态。

■ 容许的调度： 若它是一个容许执行的调度。

§ 2.1.1 系统

3.同步系统

在同步模型中，处理器按锁步骤(lock-step)执行：

执行被划分为轮，每轮里，①每个处理器能够发送一个msg到每个邻居，这些msg被传递。②每个处理器一接到msg就进行计算。

虽然特殊的分布系统里一般达不到，但这种模型对于设计算法非常方便，因为无需和更多的不确定性打交道。当按此模型设计算法后，能够很容易模拟得到异步算法。

- 轮：在同步系统中，配置和事件序列可以划分成不相交的轮，每轮由一个传递事件(将outbuf的消息传送到信道上使outbuf变空)，后跟一个计算事件(处理所有传递的msg)组成。

§ 2.1.1 系统

- 容许的执行：指无限的执行。
因为轮的结构，所以
 每个处理器执行无限数目的计算步，
 每个被发送的msg最终被传递
- 同步与异步系统的区别
 在一个无错的同步系统中，一个算法的执行只取决于初始配置
 但在一个异步系统中，对于相同的初始配置及无错假定，因为处理器步骤间隔及消息延迟均不确定，故同一算法可能有不同的执行。

§ 2.1.2 复杂性度量

■ 分布式算法的性能：

❖ 消息复杂度

❖ 时间复杂度

❖ 空间复杂度

❖ 性能衡量：最坏性能、期望性能

■ 终止：假定每个处理器的状态集包括终止状态子集，每个的 p_i 的转换函数对终止状态只能映射到终止状态

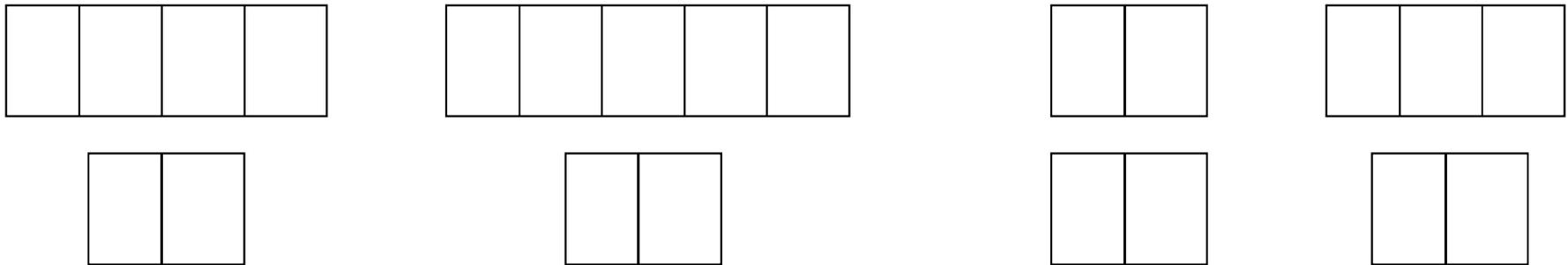
当所有处理机均处于终止状态且没有msg在传输时，称系统(算法)已终止。

§ 2.1.2 复杂性度量

■ 算法的msg复杂性(最坏情况): 算法在所有容许的执行上发送msg总数的最大值(同步和异步系统)

■ 消息复杂度度量

❖ 消息复杂度: 消息总数/消息中总的位数长度



❖ 消息总数: 4/4

❖ 消息位数总长度(位复杂度): 14/8

§ 2.1.2 复杂性度量

■ 时间复杂度

- ①同步系统：最大轮数，即算法的任何容许执行直到终止的最大轮数。
- ②异步系统：假设：
 - ①节点计算任何有限数目事件的时间为0；
 - ②一条消息发送和接收之间的时间至多为1个时间单位，定义为：所有计时容许执行中直到终止的最大时间。

■ 计时执行(timed execution)

指：每个事件关联一个非负实数，表示事件发生的时间。时间起始于零，且须是非递减的。但对每个单个的处理器而言是严格增的。

若执行是无限的，则执行的时间是无界的。因此执行中的事件可根据其发生时间来排序

不在同一处理器上的多个事件可以同时发生，在任何有限时间之前只有有限数目的事件发生。

§ 2.1.2 复杂性度量

■ 消息的延迟

- ❖ 发送msg的计算事件和处理该msg的计算事件之间所逝去的时间
- ❖ 它主要由msg在发送者的outbuf中的等待时间和在接收者的inbuf中的等待时间所构成

■ 异步算法的时间复杂性

定义中，每个msg延时至多为1，但实际中，至多1个时间单位会很难计算，因此修改假设：

- ①一条消息发送和接收之间时间恰好为1个时间单位
- ②一条消息发送和接收之间时间介于 α 和1之间($0 < \alpha < 1$)
- ③假设消息传递的延迟满足某种概率分布，并由此来计算

§ 2.1.3 伪代码约定

在形式模型中，一个算法将根据状态转换来描述。但实际上很少这样做，因为这样做难于理解。

实际描述算法有两种方法：

- ①叙述性：对于简单问题
- ②伪码形式：对于复杂问题

§ 2.1.3 伪代码约定

- 异步算法：对每个处理器，用中断驱动来描述异步算法。

在形式模型中，每个计算事件1次处理所有输入缓冲区中的msgs。而在算法中，一般须描述每个msg是如何逐个处理的

异步算法也可在同步系统中工作，因为同步系统是异步系统的一个特例。

一个计算事件中的局部计算的描述类似于顺序算法的伪代码描述。

- 同步算法：逐轮描述

- 伪代码约定：

—在 p_i 的局部变量中，无须用 i 做下标，但在讨论和证明中，加上下标 i 以示区别。

—“//”后跟注释

§ 2.2 生成树上的广播和汇集

■ 为什么广播和汇集算法

信息收集（敛播/汇集）及分发(广播)是许多分布式算法的基础。故通过介绍这两个算法来说明模型、伪码、正确性证明及复杂性度量等概念。

■ 为什么生成树上？

由于分布式系统中，每个节点并不知道全局拓扑状态，但某些算法需要在特定的结构下才能达到最优。例如：广播/敛播在树结构下才能达到消息复杂度最优，因此构造生成树是必要的，且是其他算法的基础。

§ 2.2 生成树上的广播和汇集

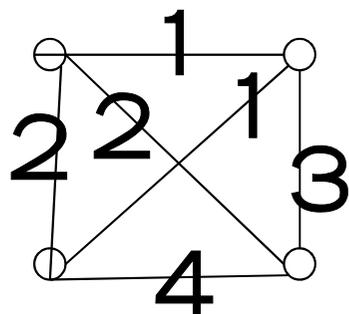
■ 生成树

一个无向连通图 G 的生成树(Spanning Tree)是指满足下列条件的 G 的子图 T :

- ① G 和 T 具有相同的顶点数;
- ② 在 T 中有足够的边能够连接 G 的所有顶点且不出现回路。

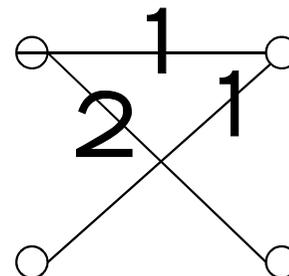
■ 最小生成树

如果图的每一条边都指定有一个权，那么所有的边权最小的生成树，就成为最小代价生成树(Minimum Cost Spanning Tree, MCST)，简称最小生成树(MST)。



生成树一共有16棵

最小生成树



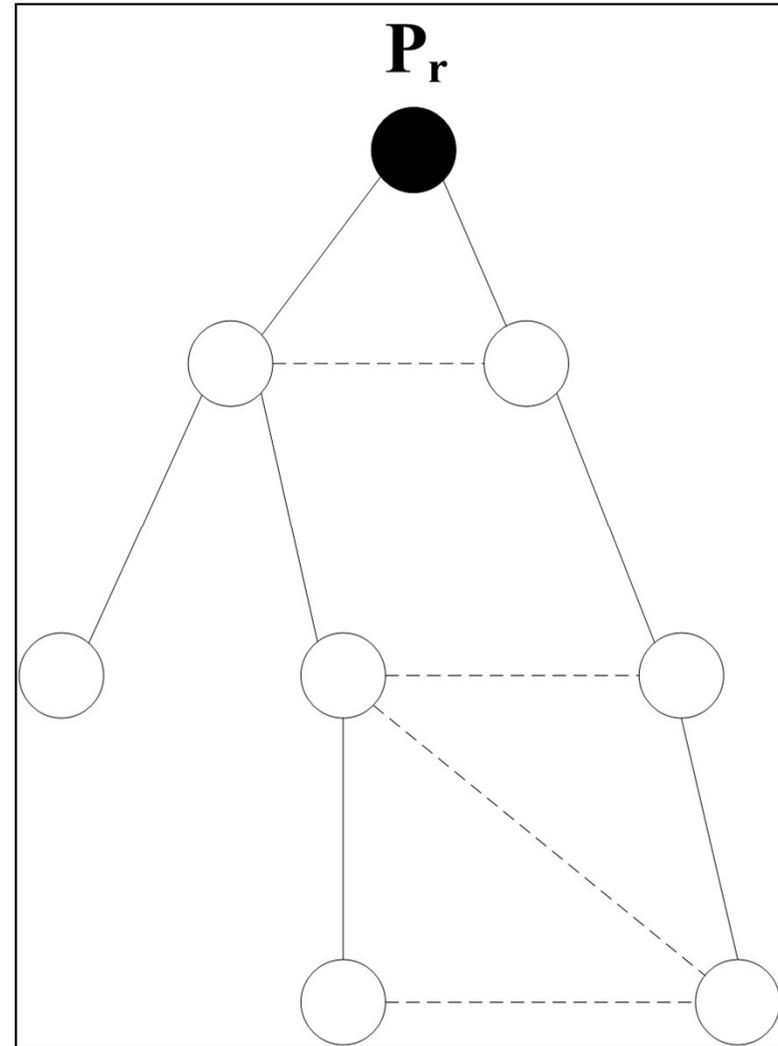
§ 2.2 生成树上的广播和汇集

§ 2.2.1 广播

(Broadcast)

假定网络的生成树已给定。某处理器 p_r 希望将消息 M 发送至其余处理器。

假定生成树的根为 p_r ，每个处理器有一个信道连接其双亲(p_r 除外)，有若干个信道连接其孩子。



§ 2.2.1 广播

- 根 p_r 发送 M 给所有孩子。(a)
- 当某节点收到父节点的 M 时, 发送 M 到自己的所有孩子(b)。

§ 2.2.1 广播

1. 伪码算法

Alg2.1 Broadcast

p_r : //发动者。假设初始化时M已在传输状态

1. upon receiving no msg: //pr发送M后执行终止
2. terminate; //将terminated置为true。

p_i ($i \neq r, 0 \leq i \leq n-1$):

3. upon receiving M from parent:
4. send M to all children;
5. terminate;

2. 用状态转换来分析算法

■ 每个处理器 p_i 包含状态

- 变量 $parent_i$: 表示处理器 p_i 双亲节点的标号或为nil(若 $i=r$)
- 变量 $children_i$: p_i 的孩子节点标号的集合
- 布尔变量 $terminated_i$: 表示 p_i 是否处于终止状态

§ 2.2.1 广播

■ 初始状态

- ❖ **parent**和**children**的值是形成生成树时确定的
- ❖ 所有**terminated**的值均为假
- ❖ **outbuf_r[j]**, $j \in \text{children}_r$ 持有消息**M**, 注意**j**不是信道标号, 而是**r**的邻居号。(任何系统中, 均假定各节点标号互不相等)
- ❖ 所有其他节点的**outbuf**变量均为空。

■ **comp(i)**的结果

若对于某个**k**, **M**在**inbuf_i[k]**里, 则**M**被放到**outbuf_i[j]**里, $j \in \text{children}_i$

§ 2.2.1 广播

■ p_i 进入终止状态

将 $terminated_i$ 置为 true；若 $i=r$ 且 $terminated_r$ 为 false，则 $terminated_r$ 立即置为 true，否则空操作。

■ 该算法对同步及异步系统均正确，且在两模型中，msg和时间复杂度相同。

■ Msg复杂度

无论在同步还是异步模型中，msg M在生成树的每条边上恰好发送一次。

因此，msg复杂性为 $n-1$ ，即 $O(n)$ 。

时间复杂度为 h ，即 $O(h)$ ，其中 h 为生成树的高度。

§ 2.2.1 广播

输入：根节点上的消息<m>

输出：每个节点都收到消息<m>

Code for P_i

Begin

while (receiving no message) do

(1) if $i=r$ then \ \此节点为根节点

(1.1) send <m> to all children

(1.2) terminates

end if

end while

while (receiving <m> from P_j) do

(1) send <m> to all children

(2) terminates

end while

end

说明：

本算法中While并不代表循环，而是代表满足条件时，节点所做的动作

§ 2.2.1 广播

■ 时间复杂性:

①同步模型: 时间由轮来度量。

Lemma2.1 在同步模型中, 在广播算法的每个容许执行里, 树中每个距离 p_r 为 t 的处理器在第 t 轮里接收消息 M 。

pf: 对距离 t 使用归纳法。

归纳基础: $t=1$, p_r 的每个孩子在第1轮里接收来自于 p_r 的消息 M

归纳假设: 假设树上每个距 p_r 为 $t-1 \geq 1$ 的处理器在第 $t-1$ 轮里已收到 M 。

归纳步骤: 设 p_i 到 p_r 距离为 t , 设 p_j 是 p_i 的双亲, 因 p_j 到 p_r 的距离为 $t-1$, 由归纳假设, 在第 $t-1$ 轮 p_j 收到 M 。由算法描述知, 在第 t 轮里 p_i 收到来自于 p_j 的消息 M

Th2.2 当生成树高度为 d 时, 存在一个消息复杂度为 $n-1$, 时间复杂度为 d 的同步广播算法

§ 2.2.1 广播

② 异步模型

Lemma 2.3 在异步模型的广播算法的每个容许执行里，树中每个距离 p_r 为 t 的处理器至多在时刻 t 接收消息 M 。

pf: 对距离 t 做归纳。

对 $t=1$ ，初始时， M 处在从 p_r 到所有距离为1的处理器 p_i 的传输之中，由异步模型的时间复杂性定义知， p_i 至多在时刻1收到 M 。

$p_i \in \{\text{距 } p_r \text{ 为 } t \text{ 的处理器}\}$ ，设 p_j 是 p_i 的双亲，则 p_j 与 p_r 的距离为 $t-1$ ，由归纳假设知， p_j 至多在时刻 $t-1$ 收到 M ，由算法描述知， p_j 发送给 p_i 的 M 至多在 t 时刻到达。

Th 2.4 同Th 2.2

下次继续!